Manual de referencia para el SINCLAIR QL

Tim Hartnell





Manual de referencia para el SINCLAIR QL

Tim Hartnell



CHIQUINQUIRA, 28 (COCUY) - 28033 MADRID
TELEF.: 764 50 95

Manual de referencia para el SINCLAIR QL

Tim Hartnell

Andrew Nelson

Tìtulo de la obra original:
QL Handbook
por Tim Hartnell

Traducido por:
M.C.Dopazo

Primera ediciòn en español:
© RA-MA
c/ Chiquinquirà, 28 (Cocuy)
28033 Madrid

Primera impresiòn Febrero 1985

Los programas de este libro se han incluido por su valor didàctico. Aunque se ha tenido el mayor cuidado , el editor no se responsabiliza de los posibles errores que se hayan introducido.

Algunas partes de este libro estàn contenidas en el volumen titulado "Explorando el Sinclair QL - Una introducciòn al SuperBASIC".

Cualquier consulta referida a este libro o a su contenido deberà dirigirse a Editorial RA-MA.

Gráficas J. C. J., S. A. - D.L.: GU-53/85
I.S.B.N.: 84-86381-02-9

Contenido

Introducción

El QL es una máquina apasionante desde el punto de vista "hardware". Pero su lenguaje interno es, en mi opinión, una de las partes más importantes.

El SuperBASIC nos proporciona, por primera vez en un microordenador popular, un lenguaje sumamente potente a la vez que elegante. Es éste un lenguaje que le alienta a la "programación estructurada"(un término que, si no entiende su significado en este momento, no tardará en comprender); un lenguaje, en definitiva, que le ayudará a programar limpia, clara y correctamente. Las técnicas que irà aprendiendo trabajando con el SuperBASIC le ayudarán cuando en el futuro quiera aprender otros lenguajes de alto nivel como: Pascal, FORTH, Modula 2 y otros.

En este libro, hemos intentado hacer el SuperBASIC tan fácil de aprender como simple de apreciar y aplicar. En un principio hemos asumido que sus conocimientos en el campo de la programamción son mínimos por lo que, aunque el QL sea su primer ordenador, conseguirá superar fácilmente los primeros pasos e introducirse sin problemas en la programación estructurada. Si ya tiene experiencia con otras máquinas, comprobará que los conocimientos que ya posee pueden adaptarse y extenderse fácilmente aprovechando las ventajas que le ofrece la potencia y claridad del SuperBASIC.

El libro está dividido en varias secciones. En las primeras veremos los componentes fundamentales del SuperBASIC -los "ladrillos" con los que usted construirá sus programas- y como funcionan aisladamente. Estos "ladrillos" los usará posteriormente para construir "estructuras" a medida que desarrolla los programas de utilidades y juegos que le iremos proponiendo a lo largo del libro y que tienen por si mismos una gran utilidad. Estos no son solamente programas de "juguete" que, aùnque le puedan enseñar un par de cosas acerca del uso de alguna palabra o concepto concretos, no son particularmente interesantes, sino que la mayor parte de ellos tendrán para

usted un gran valor intrínseco que hará de su uso verdaderas unidades de aprendizaje.

A medida que vaya realizando los programas principales, iremos viendo distintas formas y trucos de programación, examinando otras facilidades como el sonido y los gráficos. Probaremos a hacer música, definir nuestros propios caracteres y muchas otras cosas que el QL es capaz de hacer.

Una vez que sepa manejar sus "ladrillos" con soltura y se haya familiarizado con las formas principales de la programación estructurada, pasaremos a conocer otros dos lenguajes que pueden ser emulados fácilmente con el SuperBASIC. En primer lugar construiremos una "máquina FORTH" que nos permitirá estudiar la "notación Polaca inversa", tal como se usa en el FORTH (y en algunas calculadoras que trabajan con "stack" [pila]). Este "FORTH" es sólamente una pequeña punta de iceberg de este potente lenguaje, pero nos servirá para introducirnos en él.

El segundo "lenguaje" que veremos será el LOGO. Como ya sabe, el QL viene equipado con "gráficos de tortuga" como parte de su vocabulario. Sin embargo, estos gráficos son solo una pequeña parte del rico lenguaje LOGO. Con el programa QLogo intentaremos emular una gran parte, aúnque no todo, del LOGO real. Esto nos va a permitir programar el QL usando el vocabulario normal del LOGO y producir una gran cantidad de efectos.

Estos dos programas han sido incluidos para demostrarle tanto la potencia del SuperBASIC como para darle una herramienta versátil con la que aprender y experimentar.

Una vez que maneje con facilidad estos dos mini-lenguajes, terminaremos el libro examinando detalladamente la creación de un juego de aventuras. Esto le ayudará a desarrollar su propia versión de estos juegos tan populares y le preparará para realizar programas de alta calidad.

Las secciones del libro no están claramente separadas como podrá haber pensado en un principio. Las materias

principales que hemos mencionado sirven solamente para darle una idea de la direcciòn que sigue el libro. Como podrà comprobar a medida que lo vaya leyendo , los diferentes temas se van solapando unos con otros. Esto nos ayudarà a hacernos una idea de la naturaleza armoniosa del SuperBASIC asî como a penetrar en los conceptos generales del lenguaje y su manejo.

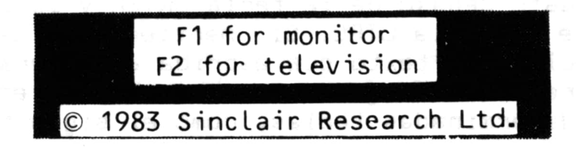
Tim Hartnell,
Londres

Capítulo Uno

Primeros Pasos en el QL

Cuando conecte por primera vez su QL, verà que la pantalla se llena brevemente de barras verticales rojas y verdes. Estas son reemplazadas posteriormente con lo que podríamos denominar "morralla de alta resoluciòn" mientras que el QL realiza sus pruebas internas e inicia la memoria.

A continuaciòn, la pantalla se pone de color negro con el siguiente mensaje en la parte inferior:



Si pulsa la tecla F1 (para indicarle que tiene un monitor conectado en el zòcalo marcado RGB) la pantalla nos mostrarà un àrea rectangular que ocupa las dos terceras partes de la pantalla. La mitad izquierda de color blanco y la derecha de color rojo. Si pulsa F2 (para indicarle que tiene un televisor conectado al zòcalo marcado UHF) la pantalla se pondrà totalmente blanca.

Las Teclas màs Importantes

Hay en el QL cuatro teclas con las que debe estar familiarizado antes de comenzar a programarlo.

La tecla ENTER que tiene forma de "L al revès" y que està situadà en la tercera fila contando desde abajo en la parte derecha del teclado. Esta tecla tiene varias funciones. Se usa cuando se quiere que el QL ejecute un comando. Si ha introducido un comando directo, tal como PRINT 6 + 7, deberà pulsar ENTER para indicarle al QL que ejecute la

instrucción que le acaba de teclear (en este caso imprimir el número 13).

También se usa después de un comando como RUN para ejecutar el programa que hemos introducido en su memoria, o después de NEW para borar la memoria del ordenador. En estos dos casos, la tecla ENTER, le indica al QL que lo que queremos es que ejecute el último comando introducido.

Cuando está tecleando un programa, debe usar la tecla ENTER para que el QL acepte la línea que acaba de teclear y que aparece en la parte inferior de la pantalla, y la añada a su programa.

El QL dispone de otras dos teclas que sirven para cambiar a mayúsculas (SHIFT), éstas se encuentran situadas, una a la derecha justo debajo de la tecla ENTER y la otra la izquierda sobre la tecla CTRL. Si mantiene pulsada una de estas dos teclas, el texto que escriba aparecerá en mayúsculas (o aparecerá el signo que se encuentra en la parte superior de determinadas teclas, como el signo $ en el 4).

Para evitar tener que mantener pulsada la tecla SHIFT cada vez que quiera escribir un texto en mayúsculas, puede pulsar la tecla CAPSLOCK (sobre la tecla SHIFT de la izquierda) con lo que todas las letras que introduzca desde el teclado aparecerán en mayúsculas (a menos que pulse la tecla SHIFT que en este caso las convertirá en minúsculas). Presionando CAPSLOCK de nuevo se restaura el teclado al modo normal.

Para borrar una letra que acaba de teclear, mantenga pulsada la tecla CTRL (CONTROL) que se encuentra en la parte inferior izquierda del teclado, y al mismo tiempo retroceda el cursor con la flecha que apunta a la izquierda (justo al lado de CTRL) e irá borrando los caracteres sobre los que pase. La tecla CTRL se usa también en combinación con la tecla ALT (en la esquina inferior derecha) y otra tecla alfabética para conseguir algunas acciones específicas del ordenador.

Líneas y Sentencias

Una sentencia de SuperBASIC le dice al QL que ejecute una acción concreta, ya sea inmediatamente o cuando sea ejecutado el programa que la contiene. Si la sentencia empieza por un número de línea, el QL sabe que ésta forma parte de un programa, y no lo ejecutará hasta que ejecute el programa con la sentencia RUN.

Sin embargo, si la sentencia no lleva número delante, el QL asume que lo que quiere es ejecutarla tan pronto como pulse la tecla ENTER.

Por lo tanto, si usted teclea PRINT "HOLA" y luego pulsa ENTER, la palabra HOLA se imprimirá en la pantalla. Pero si teclea 10 PRINT "HOLA" y a continuación pulsa ENTER, la línea se trasladará a la parte superior de la pantalla para asegurarle que la línea de programa ha sido aceptada. En este momento deberá teclear RUN y pulsar ENTER para que se imprima la palabra HOLA en la pantalla.

Una sentencia directa destinada a su ejecución inmediata puede estar compuesta por varias sentencias simples separadas mediante dos puntos (:). Por lo tanto PRINT "HOLA" : PRINT "ADIOS" puede ser introducido como una sola sentencia directa y su resultado -cuando pulse ENTER- será la aparición en la pantalla de la palabra HOLA seguida de la palabra ADIOS.

Palabras Clave

Las palabras clave del SuperBASIC se dividen en tres grupos principales: procedimientos, funciones y comandos. El usuario puede añadir sus propias palabras clave mediante la definición de procedimientos.

Estos son los procedimientos predefinidos:

ARC, ELLIPSE, LINE, POINT, SCALE, BEEP, INPUT, PAUSE, PRINT, BAUD, CLOSE, COPY, COPY_N, DELETE, DIR, FORMAT,

OPEN, OPEN_IN, OPN_NEW, POKE, POKE_W, POKE_L, RANDOMISE, CLEAR, CONTINUE, EXEC, EXEC_W, LBYTES, LIST, LOAD, LRUN, MERGE, MRUN, NEW, RETRY, RUN, SAVE, SBYTES, SEXEC, STOP, AT, BLOCK, BORDER, CLS, CSIZE, CURSOR, FLASH, INK, OVER, PAN, RECOL, SCROLL, STRIP, UNDER y WINDOW.

Las funciones predefinidas son :

BEEPING, INKEY$, KEYROW, ABS, ACOS, ACOT, ASIN, ATAN, COS, COT, EXP, INT, LN, LOG10, SINC, SQRT, TAN, PI, CHR$, CODE, PEEK, PEEK_W, PEEK_L y RND.

Los comandos predefinidos en el QL son los siguientes:

FOR, REPeat, SELect, ON, IF THEN ELSE, DEFine PROCedure, DEFine FuNction, RETURN, DATA, END FOR, END IF, END REPeat, END DEFine, END SELect, EXIT Y NEXT.

Cuando se introducen los comandos, no es necesario teclear la palabra (o palabras) completas. Una vez que ha introducido la parte que aparece en mayúsculas (como REP por REPeat), el QL se encargará de escribir el resto.

Comandos y funciones de control de programas:

CLEAR
CONTINUE
DLINE lista de rangos
EXEC nombre
EXEC_W nombre
LBYTES nombre, dirección principio
LIST [#n,] lista de rangos
LOAD nombre
LRUN nombre
MERGE nombre
MRUN nombre
NEW
RENUM [línea [TO línea;] [principio][,incremento].
RETRY

```
RUN [nùmero de lìnea]
SAVE nombre [lista de rangos]
SBYTES nombre,direcciòn principio, longitud
SEXEC nombre,direcciòn  principio,  longitud,
                                  espacio de datos
STOP
```

Otros comandos y funciones:

```
CHR$ [codigo de caràcter]
CODE [cadena]
MODE 4/8/256/512
PEEK [direcciòn]
PEEK_L [direcciòn]
PEEK_W [direcciòn]
POKE direcciòn,octeto
POKE_L direcciòn,palabra
POKE_W direcciòn,palabra larga
RND [([m TO ]n)]
```

Modos Gràficos

Ya veremos màs adelante los modos gràficos con detalle. Sin embargo, a modo de breve introducciòn, diremos que hay dos: MODE 256 (o MODE 8) que es el de màs baja resoluciòn, con 256 "pixels" (puntos o elementos de dibujo) a lo ancho de la pantalla y MODE 512 (o MODE 4) que le da doble resoluciòn que el anterior.

El QL asumirà MODE 256 si pulsa la tecla F2 al conectarlo, para indicarle que tiene conectado un televisor, en este modo el cursor es un cuadrado intermitente de color pùrpura. Si le indica que tiene conectado un monitor, mediante la tecla F1, asumirà el MODE 512, con el cursor en forma de rectàngulo estrecho de color rojo intermitente.

Colores

El QL dispone de ocho colores (azul, rojo, magenta, verde, azul claro, amarillo, blanco y negro) cuando se trabaja en MODE 256, mientras que en MODE 512, solamente puede usar

cuatro (negro, rojo, verde y blanco).

Se pueden crear màs colores mediante el uso de diferentes combinaciones de "punteado". Disponemos de cuatro modelos de punteado (barras verticales, barras horizontales, puntos grandes de fondo y puntos pequeños de fondo) que le permiten crear un amplio espectro de colores.

Si està usando un televisor o monitor en blanco y negro, en el modo 256 podrà ver ocho escalas de gris, mientras que en el modo 512 solamente se ven cuatro. El nùmero de caracteres que pueden aparecer en la pantalla viene determinado por el modo en que se encuentre. Si està usando un monitor en modo 512, dispondrà de 25 lìneas de 84 caracteres. El formato normal para un televisor es de 37 a 60 columnas, dependiendo del programa.

El Reloj de Tiempo Real

El QL tiene un dispositivo que se encuentra en pocos ordenadores, un reloj de tiempo real. Para acceder a èl se usa el comando DATE$ (como en PRINT DATE$).

Este reloj contiene un calendario completo asì como un reloj horario en formato de 24 horas. Tiene el siguiente formato:

dia-mes-año hora-minutos-segundos

Para ajustar el reloj interno se usa el comando SDATE (o sdate, ya que el QL permite introducir comandos tanto en mayùsculas como en minùsculas o combinàndolas), como se ve a continuaciòn:

```
sdate 1984,09,19,08,20,03
```

Teclee este ejemplo y pulse luego la tecla ENTER. Espere un momento, teclee PRINT DATE$ y verà algo parecido a esto:

```
1984 Sep 19 08:20:56
```

Espere otro poco y PRINT DATE$ le darà esta otra respuesta

1984 Sep 19 08:21:33

Para ajustar la fecha o extraer parte de ella puede usar el siguiente programa:

```
10 REMark Extraer partes de la fecha
20 CLS:CLS #0
30 CSIZE 3,1
40 SDATE 1986,1,1,1,0,0
50 PRINT DATE$
60  REPeat tiempo
70   G$=DATE$
80   AT 3,6
90   PRINT G$
100 END REPeat tiempo
```

Super Reloj

Este programa, que se publicò originalmente en la revista *Quanta* del Independient QL User's Club, crea una ventana que ocupa la pantalla completa y coloca en ella la fecha y la hora. El resultado es bastante bueno, pero necesita que el programa se estè ejecutando constantemente. El efecto es mejor si le ponemos INK blanco y PAPER rojo.

```
100 REMark     Super Reloj
110 REMark    de QUANTA 1/7
120 :
130 reloj
140 :
150 DEFine PROCedure reloj
160  WINDOW #1,512,256,0,0
170  CLS
180  CSIZE 3,1:OVER 0
190 :
200 REPeat tic
210  a$=DATE$
220  AT 6,7:PRINT DAY$;" ";a$(6 TO 11
);" ";a$(1 TO 4)
```

```
230  b$=a$(13 TO 14):C$="am"
240  IF b$>12 THEN b$=b$-12:c$="pm"
250  IF b$<10 THEN b$="0" & b$
260  a$(13 TO 14)=b$
270  AT 4,9:PRINT a$(13 TO):" ";c$
280  IF INKEY$<>"" THEN EXIT tic
290 END REPeat tic
300 :
310  WINDOW #1,422,192,40,18:CLS:CSIZ
E 1,0
320 END DEFine reloj
```

Capítulo Dos

Cuidado y Carga de los Microdrives

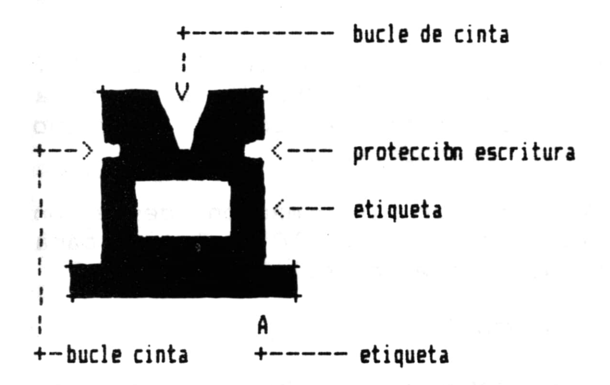
El QL usa los "microdrives" para almacenar los programas y datos que, de otra forma, se perderían al apagar el ordenador. Es esencial, en este momento, conocer algo sobre ellos para que pueda ir salvando los programas que le iremos proponiendo a lo largo de este libro.

Los cartuchos de "mocrodrive" deben ser tratados con el co.ando FORMAT para acondicionar la cinta. Para ello debe usar el comando de la siguiente forma:

```
FORMAT MVD1_nombre_del_cartucho
```

Tenga mucho cuidado con este comando, ya que si lo ejecuta sobre un cartucho que contenga información, ésta será borrada.

Estas son las partes principales de un cartucho de "microdrive":



Ya que los "microdrives" son el principal medio de almacenamiento de datos de su QL, deben ser tratados con

sumo cuidado. Es una buena norma tener copias repetidas del material màs importante, para el supuesto de que se dañe la copia principal. Cada cartucho puede contener hasta 100 k, es conveniente que separe un grupo de cartuchos como copias de seguridad, le ocuparà poco tiempo y puede serle sùmamente ùtil màs adelante.

Dentro de cada cartucho hay un bucle de cinta continua de unos 5 metros (200 pulgadas) aproximadamente, que gira dentro de una pequeña caja de plàstico a una velocidad aproximada de 2.5 km por hora (1.6 millas). Dicho de esta forma no parece excesivamente ràpido, pero significa que la cinta pasa por la cabeza de lectura a una velocidad de 8.41 metros por segundo.

Aunque la cinta es de vídeo, esto no significa que sea invulnerable y necesita ser tratada con mucho cuidado. no la toque nunca con los dedos y, si se sale del cartucho, vuelvala a su lugar con cuidado pero sin tocarla con las manos.

Puede dañar los datos de un cartucho si lo mete y lo saca repetidamente del QL sin usarlo o si lo apaga y lo enciende con èl montado.

Como regla general, tenga en cuenta que los pequeños y delicados cartuchos son bastante fràgiles y necesitan ser tratados con cuidado.

Puede conectar hasta seis "microdrives" adicionales al QL. El comando CAT(en formato CAT MVD1 o CAT "MVD2") se usa para obtener la lista de los ficheros contenidos en el cartucho montado en el correspondiente "microdrive".

El comando COPY se usa para copiar informaciòn desde un punto a otro del sistema. La palabra TO se usa para relacionar el canal de entrada con el de salida:

```
COPY MDV1 TO MDV2
```

Si quiere borrar un fichero determinado de un cartucho, debe usar el comando DELETE en la siguiente forma:

DELETE MDV1_nombre_del_fichero

Para cargar un programa desde un "microdrive" a la memoria del QL, se usa el comando:

LOAD MDV1_nombre_del_fichero

Si quiere que el programa se ejecute automaticamente una vaz cargado, use LRUN en lugar de LOAD (equivale al comando CHAIN de otros ordenadores, como el BBC). Para salvar sus programas debe usar:

SAVE MDV1_nombre_del_fichero

Vamos a realizar algunos experimentos simples para que se familiarize con los "microdrives".

Primero, seleccione un cartucho nuevo (uno que no tenga nada almacenado en su interior). Para prepararlo para que pueda ser utilizado, debe usar el correspondiente comando tecleando:

format mdv1_

Ponga el cartucho elegido en la ranura del "microdrive" que se encuentra màs cerca del centro del ordenador y, una vez que haya tecleado la lînea anteriormn pulse ENTER y espere mientras la luz roja permanexca encendida y la cinta gire dentro del cartucho, seguidamente aparecerà en su pantalla el mensaje:

217/219 sectors

Esto significa que, de los 219 sectores que contiene la cinta, usted puede usar 217 para almacenar programas. Le sugiero que practique el comando FORMAT varias veces y verà que en las siguientes veces aparecen màs sectores disponibles.

Una vez que la cinta ha dejado de girar, introduzca el siguiente programa, pulsando la tecla ENTER despuès de cada lînea:

```
10 CLS
20 PRINT "Este es un programa de prueba"
30 PRINT "en mi tonto y gris QL"
```

Ahora lo puede salvar en su "microdrive" tecleando el siguiente comando y ENTER al final:

```
save mdv1_prueba
```

Puede comprobar el contenido del cartucho con la siguiente lînea (debe pulsar ENTER despuès de esta lînea, y despuès de cada lînea de programa, pero para no cansarle diciendoselo en cada iînea, de ahora en adelante asumirè que lo hace):

```
dir mdv1_
```

En su pantalla aparecerà el siguiente mensaje:

```
216/219 sectors
prueba
```

Bueno, ahora ya tenemos el programa en el "microdrive", asî que podemos eliminarlo de la memoria del ordenador tecleando la palabra NEW. El programa desaparecerà, lo puede comprobar tecleando el comando LIST, que sirve para imprimir en la pantalla el listado del programa que hay en la memoria.

Para cargar el programa desde el "microdrive" a la memoria del ordenador, teclee lo siguiente:

```
load mdv1_prueba
```

Durante unos instantes oirà el zumbido de la cinta al girar y al cabo de unos segundos se pararà. Si tecles LIST en este momento, el programa aparecerà listado de nuevo en la pantala.

Y esto es todo lo que necesita saber por le momento para poder usar los "microdrives" satidfactoriamente.

Capítulo Tres

Conceptos del SuperBASIC

El lenguaje con el que se programa el QL se denomina SuperBASIC y ha sido desarrollado por Sinclair Research. A primera vista parece una nueva versiòn del BASIC clàsico. En cierto modo es muy similar al BASIC del Spectrum, e incluso lo puede programar como si lo fuera, pero el SuperBASIC es mucho màs potente.

Usàndolo eficientemente se aleja tanto del BASIC que, Sinclair pensò seriamente en denominarlo de otra forma, pero pensò tambièn que la gente no querrìa un ordenador que le suspusiera aprender un nuevo lenguaje para poder usarlo.

Los cuatro componentes fundamentales del SuperBASIC son:

- -los identificadores
- -la habilidad para agrupar y manipular elementos relaccionados
- -la coerciòn
- -el troceado

Para ayudarle a programar el QL, hay un gran nùmero de palabras clave que son idènticas a las de otros BASIC, pero que no son totalmente necesarias en SuperBASIC. Esto se ha hecho para facilitarle la transiciòn desde el BASIC clàsico al del QL, Estas palabras le permitemtambièen introducir programas escritos para otras màquinas con cambios mìnimos. Sin embargo, para usar a fondo el QL, le sugiero que se habitùe a reemplazar las funciones superfluas por las propias del SuperBASIC.

La sentencia GOTO puede ser reemplazada por IF, REPeat y otras, y GOSUB puede ser reemplazada por una llamada a un procedimiento (definido con DEFine PROCedure). ON...GOTO, que no estaba incluida en los anteriores ordenadores de Sinclair, se puede reemplazar en SuperBASIC con SELect, asì

como como ON...GOSUB.

Hay muchos otros comandos que le resultarán familiares y que en SuperBASIC realizan la misma función que en ls otros BASIC. Entre otras están PEEK y POKE, LIST, READ y DATA. Otras varían ligeramente, como REMark y SQRT que son màs conocidos como REM y SQR.

También hay algunos comandos que ya no funcionan, como LLIST, que se usa en la mayoría de los ordenadores para obtener el listado del programa sobre la impresora. En el QL se hace abriendo un canal a la impresora y usando posteriormente el comando LIST, que sirve para listar tanto a la pantalla como a la impresora.

En definitiva, verá como el QL puede hacer cualquier cosa que hagan otros ordenadores con el BASIC, e incluso más, aunque al principio le pueda resultar algo dificil encontrar el comando adecuado. Los comandos màs comunes, que no están soportados en el QL, incluyen LLIST, LPRINT, VAL, STR$, IN, OUT, ON ERROR e INPUT.

Como ya le he explicado, el QL lista a la impresora mediante el comando LIST# seguido del número del dispositivo, LPRINT se sustituye de la misma forma con PRINT#.

Los camandos VAL y STR$ no están incluidos porque no hacen falta, como verá en breve, cuando veamos una facilidad del QL llamada "coerción". Como seguramente sabrá, VAL cambia un número dentro de una cadena por su equivalente numérico (esto es, convertiría "1" en 1 para que se pudiera procesar como una variable numérica). STR$ hace lo contrario, cambia una variable numérica por una cadena (456 en "456"). En el QL, estos cambios se hacen automáticamente cuando son requeridos, por lo que VAL y STR$ resultan innecesarios.

Los comandos IN y OUT no son aplicables al Motorola 68008, el procesador principal del QL (tiene otro procesador, el Intel 8049, que se ocupa de controlar el teclado, generar el sonido y de la recepción del RS-232-c.
Hay también otros chips, diseñados para Sinclair, que se ocupan uno de la pantalla y la memoria y el otro de los

"microdrives", la red de àrea local y la trasmisiòn RS-232-C).

Antes de continuar con este "duro aprendizaje", vamos a ver un programa simple, pero incriblemente efectivo, que introduciremos y ejecutarmos en su QL. Teclelo tal como est 1 y sièntese comodamente, puès puede darle una o dos horas de entretenimineto. No le voy a explicar en este momento què es lo que hace cada lìnea, lo que quiero darle es algo muy conocido para que pueda ver la potencia del genio que tiene bajo su mando y, què tipo de cosas puede hacer. (El programa en sì se lo explicarè màs adelante, en la secciòn de gràficos de tortuga, como parte del *QLogo*).

```
100 REMark Graficos de tortuga
110 PAPER 0
120 CLS:CLS #0
130 PENDOWN
140 REPeat logo
150  IF RND>.68 THEN CLS
160  nota=RND
170  POINT 80,50
180  INK RND(1 TO 7)
190  tortuga=RND(3 TO 70)
200  elido=RND(25 TO 340)
210   FOR k=1 TO tortuga
220    MOVE k/.7
230    TURN elido
240    BEEP 3200,k/nota
250   END FOR k
260 IF INKEY$<>"" THEN STOP
270 END REPeat logo
```

Introduzca el programa en su QL pulsando la tecla ENTER despuès de cada lìnea, teclee el comando RUN y prepàrese para disfrutar del programa. Le sugiero que lo ejecute en una habitaciòn oscura, para que sea màs efectivo. El programa se puede parar en cualquier momento pulsando una tecla cualquiera.

Nùmeros, Nùmeros, Nùmeros

Si sus primeros pasos en el mundo de los ordenadores los diò en un ZX80, como lo hice yo, quizàs recordarà que el mayor nùmero que se podìa obtener era el 32767 y el menor era -32767 (o -32766 en algunas circunstancias). El QL usa un rango que puede resultar confuso al principio. El lìmite para los enteros es tambien -32767 a +32767 y las variables que los contienen se identifican con un nombre seguido del signo del tanto por ciento (%), por ejemplo A%, sin embargo el rango de los nùmeros de coma flotante se alarga casi infinitamente.

El rango de los nùmeros de coma flotante se extiende desde -10 elevado a la potencia -615 hasta 10 elevado a la potencia 615, lo que significa que puede tener mas/menos 10 elevado a -675 decimales significativos. El QL toma por defecto este nùmero de digitos decimales significativos en el momento de encenderlo.

Las cadenas de caracteres pueden tener hasta 32768 caracteres de longitud. Un poco mas adelante hablaremos con mas detalle de las cadenas.

Los anteriores ordenadores de Sinclair tenìan un lìmite entre 1 (o 0 haciendo un POKE) y 9999 para los nùmeros de lìnea que preceden a las sentencias de un programa. El rango del QL es desde 1 a 32767.

Ejecutando Programas

Cuando, en el QL, teclea el comando RUN y pulsa ENTER, se limpian las variables (las numèricas a cero y las cadenas a cadenas vacìas). Para evitar esto puede usar el comando GOTO 1, aunque no haya lìnea 1 en su programa.

Si desea comenzar la ejecuciòn de un programa en una lìnea especìfica, puede usar GOTO n (donde n es el nùmero de la lìnea) si quiere preservar las variables, o RUN n cuando las variables han de ser limpiadas, pero en cualquiera de los dos casos, la ejecuciòn comenzarà en la lìnea n, o en la inmediata superior si n no existe.

Si por alguna razòn interrumpe la ejecuciòn del programa, puede continuar la ejecuciòn desde el punto de interrupciòn con el comando CONTINUE.

Usando el Editor de Pantalla

El comando EDIT se usa, obviamente , para editar una lìnea del programa en la parte inferior de la pantalla y poder modificarla. Puede llamar al editor tecleando el comando:

EDIT nnn (donde nnn es el nùmero de lìnea)

Si no especifica nùmero de lìnea, el ordenador tomarà la lìnea de nùmero màs bajo del programa. La lìnea se irà resaltando a medida que es modificada para indicar que el procedimiento interno de detecciòn de errores de sintaxis del QL, no ha procesado aùn la lìnea.

Una vez que haya pulsado ENTER, para indicar que ya ha terminado de teclearla, el QL comprobarà la sintaxis de la lìnea. Si està libre de errores la lìnea entrarà en el programa reemplazando la versiòn anterior ('pre-EDIT'). El QL es un duro capataz, y no le dejarà introducir la lìnea hasta que haya pasado satisfactoriamente este proceso de revisiòn de sintaxis.

Si està editando una lìnea que no cabe completamente en la pantalla, verà que la lìnea se desplaza hacia arriba automàticamente, a medida que mueve el cursor a lo largo de ella, para que la parte con la que desea trabajar aparezca en la pantalla.

Lo que usted teclèe se irà añadiendo delante de lo que ya existìa, que se irà desplazando a lo largo de la lìnea para dejar sitio a lo nuevo.

La version "pre-EDIT" podrìa ser:

```
190 A=INT(M/Q):PRINT A
```

Mueva el cursor justo despuès de los dos puntos y añada el nuevo material como sigue:

B=5*A:

Obtendrà una nueva lìnea que se leerà:

```
190 A=INT(M/Q):B=5*A:PRINT A
```

La parte original se desplazarà lateralmente para hacerle sitio a la nueva.

Si està trabajando en una lìnea, en modo EDIT, y cambia deidea, puede recuperar la lìnea original sin alterar, pulsando la tecla ESC.

Capítulo Cuatro

Identificadores

Como le dije al principio del libro vamos a hablar de los "ladrillos" del SuperBASIC, los conceptos y componentes fundamentales del lenguaje, antes de ver como pueden ser combinados para formar un programa SuperBASIC. El primero de estos ladrillos es el "identificador".

Un identificador puede tener hasta 255 caracteres de longitud y debe empezar siempre por una letra (o un signo & si es un identificador del sistema). Detràs de este primer caracter puede ir cualquier combinaciòn de letras y nùmeros y, muy importante, el caracter del subrayado (_). Si termina con el sìmbolo $ se entiende que es un identificador de una cadena. Si termina con % es de una variable entera.

El subrayado es particularmente ùtil cuando quiere usar identificadores que describan claramente la funciòn que identifican. Por ejemplo, todos los identificadores que siguen son vàlidos:

```
resultado
respuesta
resultado_final
fin_del_juego
efecto_2a
```

Asegùrese de no confundir el caracter de subrayado (_) con el guiòn (-), ya que ambos aparecen en la misma tecla en la fila superior al lado del cero. Sin pulsar la tecla SHIFT obtendrà el guiòn y pulsandola obtendrà el subrayado.

El QL, al igual que el Spectrum, no diferencia entre mayùsculas y minùsculas cuando se trata de identificadores. Esto viene a significar que los identificadores que aparecen a continuaciòn son idènticos, desde el punto de

vista del QL:

```
respuesta_3e
RESPUESTA_3E
Respueta_3E
Respuesta_3e
rEspuesTA_3e
```

Los identificadores son uno de los conceptos fundamentales del SuperBASIC. Verá como a medida que se acostumbre a trabajar con el lenguaje, usará los identificadores cada vez más en los programas que vaya escribiendo. Una vez que haya reconocido la claridad adicional que proporciona a sus programas el uso de cosas como los procedimientos (en lugar de GOTOs y GOSUBs), ayudado por los identificadores explícitos, se dará cuenta de la potente herramienta que es el SuperBASIC.

Aunque en algunas circunstancias, un identificador parezca que es una variable, en realidad no lo es. Un identificador es una etiqueta que pone el QL en un objeto específico que ha almacenado. El QL tiene otras piezas de información directamente relaccionadas con el nombre del identificador. Estas incluyen la naturaleza de lo que está identificando (una cadena? un número de coma flotante?), el ejemplo específico de lo que está alamacenado ("Jose" o 778.45) y en que lugar de la memoria está localizado este objeto.

Si usa como identificador A, el SuperBASIC puede manipular los valores relacionados con este identificador junto con cualquier valor de los identificadores relacionados B,C etc, que forman parte de la definición del identificador A.

Coerción

Como ya le dije anteriormente, Val y STR$ no son necesarios en el QL, gracias a una facilidad llamada "coerción", otro de nuestros "ladrillos".

En la mayoría de los BASICs que existen en el mercado, incluyendo los del ZX81 y el Spectrum, el introducirles una

entrada de tipo equivocado hace fallar al sistema.

Considere el siguiente programa:

```
10 REMark Coerciòn, demostraciòn
20 CLS:CLS #0
30 PRINT "Introduzca un nùmero"
40 INPUT a
50 PRINT "Ahora introduzca otro"
60 INPUT b$
70 PRINT "El total es ";a + b$
```

Este programa fallarìa en la ùltima lìnea en la mayorìa de los ordenadores, ya que intenta efectuar una suma entre un nùmero y una cadena.

Sin embargo en el QL sì se puede y realizarà la suma sin ningùn problema, siempre que responda a la segunda pregunta con un nùmero.

Supongamos que pulsa la tecla "2". Aunque su entrada se almacenarà como "2" dentro del QL (y no como 2 numèrico), el ordenador lo convertirà en 2, de forma que podrà ser sumado.

El QL realizarà la coerciòn solamente si le es posible hacerlo. Consideremos el siguiente programa, no funcionarìa en la mayorìa de los ordenadores, y que nos muestra la coerciòn actuando en modo opuesto al del primer ejemplo:

```
10 REMark Coeciòn, Demo II
20 CLS:CLS #0
30 PRINT "Cual es su nombre?"
40 INPUT a$
50 PRINT "Què edad tiene?"
60 INPUT b
70 LET c$=a$ & b
80 PRINT c$
```

Fìjese que hemos usado el ampersand (&) en la lìnea 50, en vez del signo màs (+). Esto le indica al QL que quiere trabajar con cadenas en vez de con nùmeros. (Observe

también que el LET de la línea 50 es opcional.)

Cuando lo ejecute en el QL, verá lo siguiente:

```
Cual es su nombre?
Andrès
Que edad tiene?
22
Andrès22
```

La última línea es el resultado de añadir las dos cadenas juntas. Por supuesto, su edad - en este caso - fuè aceptada y almacenada por el QL como un número y despuès coercitada dentro de una cadena de forma que pudiera ser añadida a la otra. La palabra "concatenación" se usa para describir la adición de dos cadenas.

La coerción trabajará en todos los casos siguientes:

```
LET a="345" + 96
      (resultando 441)
LET a="13"+"56"+"99"
      (resultando 168)
LET a$=245 & 245
      (resultando "245245")
LET a$="345" & 96
      (resultando "34596")
```

Cuando estè añadiendo dos números de coma flotante y asignàndolos a una variable de coma flotante (cualquier combinación de letras y números, empezando por una letra, tales como A8976 o BNGS5 o BN77BN), no es necesaria la coerción y no sucede nada.

De modo similar, si està asignando el resultado de la adición de dos números de coma flotante a una variable entera (cualquier combinación de letras y números, empezando por una letra y terminando por el signo %, como A786%, B% o BHJHJ%), los dos números de coma flotante no son coercitados, pero el resultado de la adición es almacenado como un entero.

El ampersand (&) y el signo màs (+) deben usarse con sumo cuidado. Cuàl piensa que serà el resultado de estos dos programas cuando los ejecute en el QL?

PROGRAMA A:

```
10 REMark Coerciòn A Demo
20 CLS:CLS #0
30 a$="6754.65"
40 b=152.76
50 c$=a$ + b
60 PRINT c$
```

PROGRAMA B

```
10 REMark Coerciòn B Demo
20 CLS:CLS #0
30 a$="6754.65"
40 b=152.76
50 c$=a$ & b
60 PRINT c$
```

El programa A nos devolverà "6907.41", mientras que el programa B nos devolverà "6754.65152.76". En el primer programa el QL coercita el "6754.65" como el nùmero de coma flotante 6754.65 dejando B como el nùmero de coma flotante 152.76 y los añade numèricamente despuès de convertir la respuesta de coma flotante en una cadena.

En el programa B, la primera cadena permanece como tal, y el 152.76 es coercitado como "152.76". Despuès las dos cadenas son concatenadas. El signo + dice al QL que debe tratar las variables como nùmeros y el signo & indica que las debe tratar como cadenas.

Capítulo cinco

Operadores

Vamos a añadir ahora los "operadores" a nuestro almacen de "ladrillos" del QL. Los operadores realizan operaciones, como el signo màs entre dos nùmeros, suma esos nùmeros. El QL puede usar una gran variedad de operadores:

=	igual	numèrico	- igual lògico
		cadena	- comparaciòn tipo 2
==	equivalente	numèrico	- "casi igual"
		cadena	- comparaciòn tipo 3
+		numèrico	- suma
-		numèrico	- resta
/		numèrico	- divisiòn
*		numèrico	- multiplicaciòn
<	menor que	numèrico	- menor que
		cadena	- comparaciòn tipo 2
>	mayor que	numèrico	- mayor que
		cadena	- comparaciòn tipo 2
<=	menor que o igual	numèrico	- menor o igual que
		cadena	- comparaciòn tipo 2
>=	mayor que o igual	numèrico	- mayor o igual que
		cadena	- comparaciòn tipo 2
<>	no igual	numèrico	- no igual a
		cadena	- no igual comparaciòn tipo 3

&	ampersand	cadena	- concatenación
&&		modo bit	- AND
¦¦		modo bit	- OR
^^		modo bit	- OR exclusivo
~~		modo bit	- NOT
OR		lògico	- OR
AND		lògico	- AND
XOR		lògico	- OR exclusivo
NOT		lògico	- NOT
MOD		numèrico	- mòdulo
DIV		entero	- dividir
^		flotante	- elevar a potencia
(^)		entero	- elevar a potencia
-		menos unitario	
+		màs unitario (NOP)	

Imagine que su QL ejecuta el siguiente fragmento de programa:

```
10 CLS:CLS #0
20 a=16+8*2
30 PRINT a
```

Cuando lo ejecute obtendrà como respuesta 32 (16 mas 8 veces 2). Sin embargo, si la lìnea 10 fuera cambiada por...

```
10 CLS:CLS #0
```

```
20 a=(16+8)*2
30 PRINT a
```

...el QL nos daría como respuesta 48 (dos veces la suma de 16 más 8).

La diferencia, por supuesto, está en los paréntesis. En la segunda versión de la línea 10 los paréntesis aseguran que 16 y 8 sean sumados antes de que tenga lugar la multiplicación. En el primer ejemplo la multiplicación se realiza en primer lugar.

Llamamos 'orden de prioridad' la que determina el orden en que se efectúan las operaciones aritméticas.

He aquí el orden de prioridades en el QL (de mayor a menor):

más o menos unitario
concatenación de cadenas
exponenciación
multiplicación y división (división de enteros y módulos)
suma y resta
comparación lógica
NOT
AND
OR y XOR

Usted puede incluir operadores lógicos (<, >, = y otros) dentro de una expresión. Si lo hace y la expresión es "cierta" (como A=3:PRINT A=A) el ordenador le devolverá uno. Si la expresión es falsa (como en PRINT 3<>3) el QL le devolverá cero.

Capítulo Seis

Cadenas

El QL -como otros ordenadores que utilizan BASIC- trata con dos clases de informaciòn: nùmeros y cadenas. Una variable o un identificador de cadena, se diferencian de lo numèricos porque la variable de cadena termina con el signo del dòlar. Como verà en el capìtulo de matrices, una matrìzde cadena usa tambièn el signo del dòlar para indicarque se trata de en vez de nùmeros.

Primero trataremos las cadenas y hablaremos de la coerciòn. Veremos que el QL puede manejar, en ciertas circunstancias, la informaciòn numèrica t de cadenas de una forma muy inteligente, cambiando la naturaleza de los datos para acomodarse a lo que requieran las circunstancias.

CHR$ y CODE

Este par de palabras realizan funciones complementarias. CHR$ (lamada 'char-dòlar' o 'character-string') transforma un nùmero en su caràcter equivalente.

Esto es, PRINT CHR$(46) imprimirà un punto. El còdigo ASCII del caràcter "." es 46. Si quiere obtener el còdigo de un caràcter, solo le tiene que decir al QL PRINT CODE ("n"), con el caràcter en el lugar de la n.

Teclee lo siguiente seguido de ENTER:

```
PRINT CODE("B")
```

El ordenador nos darà el nùmero 6. PRINT CODE "BATALLA" tambièn nos darà 66 ya que CODE toma solamente el primer elmento de la cadena. Introduzca el siguiente programa, pero antes de ejecutarlo trate de adivinar cual serà la respuesta:

```
10 TEST$="BANANA"
20 CLS:CLS #0
30 PRINT CODE(TEST$)
```

Una vez màs tenemos 66, ya que el QL evalùa TEST$ como si fuera "BANANA" (que es el valor que le ha asignado en la lìnea 10). Observe que no necesita las comillas rodeando el nombre de la cadena, para hacer que el ordenador le devuelva su còdigo.

Cada nùmero, letra, signo y sìmbolo que puede escribir el QL, tiene su propio còdigo asignado. He aquì algunos de los còdigos que usa el QL en su juego de caracteres:

32	! 33	" 34	# 35	
$ 36	% 37	& 38	' 39	(40
) 41	* 42	+ 43	, 44	- 45
. 46	/ 47	0 48	1 49	2 50
3 51	4 52	5 53	6 54	7 55
8 56	9 57	: 58	; 59	< 60
= 61	> 62	? 63	@ 64	A 65
B 66	C 67	D 68	E 69	F 70
G 71	H 72	I 73	J 74	K 75
L 76	M 77	N 78	O 79	P 80
Q 81	R 82	S 83	T 84	U 85
V 86	W 87	X 88	Y 89	Z 90
[91	\ 92	] 93	^ 94	_ 95
£ 96	a 97	b 98	c 99	d 100
e 101	f 102	g 103	h 104	i 105
j 106	k 107	l 108	m 109	n 110
o 111	p 112	q 113	r 114	s 115
t 116	u 117	v 118	w 119	x 120
y 121	z 122	{ 123		124	} 125

También incluye gran variedad de sìmbolos matemàticos y letras propias de otros idiomas europeos, en el caso del castellano tenemos:

à 139 è 143 ì 147 ò 150 ù 153
ñ 136 Ñ 168

Troceado

Las cadenas pueden ser troceadas. Ejecute el siguiente programa:

```
10 REMark Troceado de Cadenas
20 CLS:CLS #0
30 a$="jugador de casino"
40 PRINT a$
50 PRINT a$(1)
60 PRINT a$(2 TO 4)
70 PRINT a$(5 TO)
80 PRINT a$(1 TO 9)
```

La salida que produce es:

```
jugador de casino
j
uga
dor de casino
jugador d
```

La primera -jugador de casino- no es ninguna sorpresa. Despuès de todo hemos asignado la cadena "jugador de casino" a a$, en la lìnea 30, asì que ya esperàbamos que se imprimiera entera en la lìnea 40.

Pero, què ha sucedido en la lìnea 50?. el manejo de cadenas de Sinclair nos permite tratar las partes de una cadena separadamente, especificando la secciòn de la cadena que queremos usar mediante un nùmero o nùmeros entre parèntesis, a continuaciòn del identificador de la cadena.

La lìnea 60, a$(2 TO 4), produce "uga" como salida. El mecanismo de troceado de cadenas toma en este caso el primer nùmero entre parèntesis (que es el 2) y saca el trozo de la cadena desde el caràcter nùmero 2, hasta el siguiente nùmero de caràcter especificado (en este caso el 4). Si no existe un segundo nùmero entre parèntesis, como ocure en la lìnea 70, el ordenador se va hasta el final de la cadena produciendo, como podemos ver en su correspondiente salida, "dor de casino". Observe que el

espacio entre las dos palabras, cuenta como un caràcter.

Quê longitud tiene una cadena?

La funciòn LEN nos proporciona la longitud de una cadena, como puede ver en el siguiente programa:

```
10 REMark LEN demostraciòn
20 CLS:CLS #0
30 REPeat ciclo
40  INPUT "Cual es su nombre? ";nom$
50  IF nom$<>"" THEN EXIT ciclo
60 END REPeat ciclo
70 PRINT \\"Bien, ";nom$;", su nombre
tiene"
80 PRINT LEN(nom$);" caracteres"
90 PRINT \\"'salsa' tiene ";LEN("sals
a")
100 PRINT " caracteres"
```

NOTA: A lo largo de este libro hemos flanqueado las cadenas con comillas dobles ("). Esto no es necesario en el QL, ya que acepta tambièn la comilla simple ('). Hemos evitado el mencionarlo antes, porque hay pocos BASIC que lo usen y preferimos dejar la comilla simple para usarla como apòstrofo, cuando la necesite dentro de una cadena. Sin embargo, usted puede usarla si lo prefiere.

Jugando con el Hombre-Cadena

Vamos a ver nuestro primer programa de juego del libro, para enseñarle un poco màs sobre la manipulaciòn de cadenas. El HOMBRE-CADENA es una versiòn del viejo AHORCADO. El ordenador escoje una palabra (entre las listadas en las lìneas 400 a 420) y le reta a adivinarla. Observe como LEN, como acabamos de ver, se usa en varias partes del programa (como en las lìneas 200 t 290).

Todo lo que necesita hacer con el programa es introducirlo y seguir las instrucciones en la pantalla. No necesita pulsar ENTER despuès de introducir la letra. Observarà en

el listado del programa que, algunas lineas tienen un espacio entre el número y el texto de la linea. Esto se ha hecho para que el programa sea más fácil de leer y para dividirlo en secciones fácilmente identificables.

Cuando haya jugado varias veces al HOMBRE-CADENA, cambie las palabras en las lineas 400 a 420 por las que usted quiera, para hacer el programa más variadom

```
10 REMark   HOMBRE-CADENA
20 REMark For Scott Vincent
30 REMark *******************
40   RESTORE
50   RANDOMISE
60   BORDER 2,3
70  CLS:CLS #0
80 AT 1,13:PRINT "HOMBRE-CADENA"
90 AT 2,13ZPRINT "-------------"
100   n=INT(RND*12)+1
110  FOR l=1 TO n
120   READ B$
130 END FOR l
140 a$=""
150 FOR l=1 TO LEN(B$)
160   a$=a$&"_"
170 END FOR l
180 REMark ******************
190 AT 15,1:PRINT"Intentos Fallados::
X=15:Y=20
200 AT 5,20-LEN(a$)/2:PRINT a$
210 IF a$=B$ THEN GO TO 360
220 AT 10,5:INPUT"Digame una letra: "
;
230 i$=INKEY$:IF i$<"A" OR i$>"z" THE
N GO TO 230
240 PRINT i$
250     PAUSE 64
260 IF CODE(i$)>CODE("Z") THEN i$=CHR
$(CODE(i$)-32)
270 AT 10,18:PRINT " "
```

```
280 V=0
290 FOR l=1 TO LEN(B$)
300   IF B$(l)=i$ THEN a$(l)=i$:V=1
310 END FOR l
320 IF V=1 THEN GO TO 200
330 AT X,Y:PRINT i$;",":Y=Y+2
340 IF Y=36 THEN Y=2:X=X+1
350 GO TO 200
360 FOR l=100 TO 1 STEP -1
370   BEEP 30,1
380 END FOR l
390 REMark ******************
400 DATA "MICROORDENADOR","SINCLAIR",
"JUEGOS","INNOVACION"
410 DATA "ENORME","ARTIFICIAL","SATIS
FACION","SOMBRA"
420 DATA "PASTAS","IMPERIO","GALLETAS
","URGENCIA"
```

Capítulo Siete

Matrices

Las matrices forman otro de los "ladrillos" esenciales de nuestro arsenal de programaciòn. Las matrices se usan cuando se quiere preparar una lista de datos (nùmeros o cadenas) a los que se pueda acceder refirièndose a su posiciòn dentro de la lista.

Las matrices se definen usando la sentencia DIM.

Una matriz es un grupo de espacios de memoria reservados dentro del QL. Nos podemos referir a cualquier elemento de la matriz dando el nombre de la matriz y un subìndice que indica la posiciòn del elemento dentro de la misma. Para producir en el QL una matriz que pueda contener cuatro nùmeros, introduzca la siguiente lìnea (asumiendo que el nombre de la matriz es HOLD):

```
10 DIM hold(3)
```

Si quiere una matriz que contenga 45 elementos numèricos, deberà usar una sentencia como èsta:

```
10 DIM nombre_de_la_matriz(44)
```

Como puede ver el nùmero entre parèntesis es uno menos que el nùmero de elementos que queremos. Esto se debe a que el primer elemento de una matriz es referido como elemento 0.

Imaginemos que ha preparado una matriz de cuatro elementos con la sentencia:

```
DIM a(3)
```

Siempre que dimensione una matriz numèrica como èsta, cada elemento contendrà un cero. Por lo tanto, el siguiente

programa...

```
10 DIM a(3)
20 CLS:CLS #0
30 PRINT a(0)
40 PRINT a(1)
50 PRINT a(2)
60 PRINT a(3)
```

...producirà...

```
0
0
0
0
```

Para llenarla debe proceder así:

```
10 DIM a(3)
20 CLS:CLS #0
30 a(0)=10
30 a(1)=9.97
40 a(2)=898273
50 a(3)=65.8781
```

Para referirnos al elemento que contiene 898273 en la matriz (o lo que es lo mismo, para referirnos al 898273), todo lo que tenemos que hacer es referirnos a A(2). Un programa puede manipular A(2) tan fàcilmente como cualquier otra variable.

La matriz que acabamos de ver es una matriz uni-dimensional que puede para contener nùmeros.

El QL, como la mayoría de los ordenadores, le permite definir matrices de màs de una dimensiòn. Estas matrices se denominan matrices multi-dimensionales. No hay límite para el nùmero de dimensiones que usted puede tener en una matriz en el QL (aparte de la limitaciòn impuesta por la memoria).

Cuando usted dimensiona una matriz, el QL reserva espacio

en su memoria para cada uno de sus elementos. La memoria es reservada aunque no se usen todos los elementos de la matriz, por lo que las matrices multi-dimensionales consumen memoria a una velocidad prodigiosa. Pero no se preocupe que, aunque usted ya ha cogido una buena cantidad para jugar en su QL, todavía le queda bastante como para no tener problemas.

Las matrices multi-dimensionales se definen añadiendo un segundo (o tercero o màs) nùmero despuès del primero, en una lìnea DIM, como puede ver a continuaciòn:

```
10 DIM hat(3,2)
```

Esta lìnea prepara una matriz de cuatro por tres. Puede imaginar el contenido de una matriz uni-dimensional como una lista de nùmeros, como los que puede ver a continuaciòn:

```
89788
 2314
12314
45621
```

Sin embargo, las matrices bi-dimensionales (como la HAT anterior) las puede imaginar como una tabla. Cuando se dimensiona por primera vez, todos los elementos de la matriz son - como ya save - puestos a cero. Aquì està el contenido de los elementos de HAT, justo despuès de ser dimensionada:

```
    0 1 2 3
  +--------
0 | 0 0 0 0
1 | 0 0 0 0
2 | 0 0 0 0
```

Cada elemento dentro de la matriz es referido por medio de dos subìndices. Por ejemplo, el elemento de la esquina inferior derecha de la tabla es HAT(3,2).

MATRICES DE CADENA

Las matrices que hemos visto hasta ahora han sido matrices numèricas, diseñadas para contener nùmeros.

El QL le permite definir tambièn matrices de cadenas. Estas se deben dimensionar de forma ligeramente diferente. Si usamos una sentencia como la que sigue...

```
DIM nombre$(5)
```

...podrìamos pensar que debe crear una matriz con espacio para almacenar seis cadenas completas. Sin embargo no es asì, a diferencia de la mayorìa de los otros BASIC no-Sinclair (la siguiente peculiaridad es compartida por el BASIC del ZX81 y el Spectrum), con una sentencia DIM como la precedente el QL prepara una matriz de un solo caràcter.

Solo puede ser usada como sigue:

```
10 DIM gee$(4)
20 RESTORE
30 FOR h=1 TO 4
40  READ gee$(h)
50 END FOR h3)
60 data "h","e","l","l"
```

Para almacenar palabras completas debemos darle una segunda dimensiòn con un nùmero que sea igual a la longitud de la cadena màs larga que se vaya a almacenar en dicha matriz. Para demostràrselo veamos el siguiente programa que prepara una matriz capaz de contener tres cadenas de hasta diez letras cada una:

```
10 DIM zumbo$(3,10)
20 RESTORE
30 FOR g=13
40  READ zumbo$(g)
50 END FOR g
60 DATA "jeremy rus","t hartnell","li
z north"
```

Aunque es necesario añadir un segundo número a la sentencia DIM cuando se prepara una matriz de cadenas, no es necesario usarlo cuando se refiera a una cadena. El siguiente fragmento de programa nos lo aclarará:

```
10 DIM saludo$(2,4)
20 CLS:CLS #0
30 saludo$(1)="hola"
40 saludo$(2)="yuju"
50 PRINT saludo$(1)
60 PRINT saludo$(2)
```

La salida de este programa sería...

```
hola
yuju
```

...en vez de...

```
h
y
```

...como podría esperarse.

Troceado de Matrices de Cadena

Ya vimos en el capítulo anterior, como funciona el troceado con cadenas simples. Ahora aplicaremos estos conocimientos a los elementos de las matrices de cadena. Examine el siguiente programa y, antes de ejecutarlo en su QL, intente predecir sus resultados:

```
10 DIM gatos$(5,9)
20 CLS:CLS #0
30 gatos$(3)="bill"
40 gatos$(5)="swing"
50 gatos$(0)="back"
60 gatos$(1)="logic"
70 gatos$(2)="rusty"
80 PRINT gatos$(0)(1 TO 3)
90 PRINT gatos$(1)(TO 4)
```

```
100 PRINT gatos$(2)(4 TO)
110 PRINT gatos$(3)(1)
120 PRINT gatos$(4)
130 PRINT gatos$(5)(5 TO)
```

Una vez hecha su predicciòn, introduzca y ejecute el programa en el QL, para ver si ha acertado. Si lo hizo, indica que ha cogido ràpidamente los conceptos del troceado de cadenas en el SuperBASIC.

Aquì tiene los resultados:

```
bac     - este es GATOS$(0)(1 TO 3)
logi    - este es GATOS$(1)(TO 4)
ty      - este es GATOS$(2)(4 TO)
b       - este es GATOS$(3)(1)
        - este es GATOS$(4) que no ha sido
                            asignada
g       - este es GATOS$(5)(5 TO)
```

Tambièn puede asignar trozos de una cadena a otra.

```
a$="salvaje"(3 TO 5)
```

...harà a$ igual a "lva".

Tambièn puede cambiar una sola parte de una cadena usando el troceado, como se ve en este programa:

```
10 prim$="Sinclair"
20 prim$(3 TO 5)="BBC"
30 PRINT prim$
```

Cuando ejecute este programa, su QL producirà este resultado:

```
SiBBCair
```

Esto nos muestra que los elementos 3, 4 y 5 de la cadena original ("NCL") han sido reemplazadas con la otra cadena ("BBC").

SUSTITUCION DE MATRICES

En este momento, probablemente se le habrà ocurrido que puede ser posible intercambiar partes en las matrices de cadena. Esto se puede hacer, como se ve en nuestro siguiente programa:

```
10 DIM palabra$(2,4)
20 palabra$(0)="vago"
30 palabra$(1)="pera"
40 palabra$(2)="lava"
50 palabra (1)(3 TO)=palabra$(2)(1 T
O 2)
60 palabra$(0)=palabra$(1)(4) & pala
bra$(0)
70 PRINT palabra$(0)
80 PRINT palabra$(1)
90 PRINT palabra$(2)
```

La salida de este inspirado programa es la siguiente:

```
avag
pela
lava
```

Que le ha ocurrido a nuestra 'palabra$' original?. Excepto "lava", las de las lìneas 20, 30 y 40 han sido reemplazadas. Si observamos las lìneas 50 y 60 veremos lo que ha sucedido.

La lìnea 50 pone las dos ùltimas letras de 'palabra$(1)' iguales a la primera y segunda de 'palàbra$(2)'. La lìnea 60 pone 'palabra$(0)' igual a la cuarta letra de 'palabra (1)' y despuès intenta añadir toda la 'palabra$(0)' original al final, pero no tiene suficiente sitio ya que la matriz original se dimensionò para palabras de cuatro letras. En SuperBASIC, si no hay sitio para una cadena en una matriz, èsta es truncada.

Capítulo Ocho

Primeros Pasos en la Programación Estructurada

Una de las mejores características del SuperBASIC es el estímulo que le da para construir sus programas de una manera clara y estructurada. Aunque conserva los clàsicos y poco elegantes mecanismos de bifurcaciòn (como GOTO, GOSUB y ON...GOTO /ON...GOSUB) es solamente para mantener cierto grado de compatibilidad con otros BASICs.

Ya le sugerí anteriormente que, si està familiarizado con otros BASIC, especialmente con el del Spectrum, puede empezar a programar el QL màs o menos como si fuera un Spectrum, e ir introduciendo gradualmente elementos del BASIC QL en sus programas.

Las secciones de control, bucles y bifurcaciònes de su programa son las que van a ganar màs con el SuperBASIC.

Las Estructuras

El vocabulario del QL es particularmente rico en este àrea, ofreciendole las siguientes posibilidades:

```
IF           (THEN)        ELSE      END IF
SELect_ON    (ON var)=               END SELect
FOR          (NEXT]        (EXIT)    (END FOR)
REPeat       (NEXT)        (EXIT)    END REPeat
```

Nuestros viejos amigos GOTO y GOSUB siguen estando disponibles en el QL, aunque no es recomendable su uso.

```
GOTO
GOSUB RETURN
```

```
ON exp GOTO/GOSUB lista
```

El QL le ofrece también procedimientos con variables locales:

```
DEFine PROCedure/FuNcion$ nombre [(lista de parámetros)]
LOCAL
RETURN [valor]
END DEFine
```

COMPATIBILIDAD

Empezaremos viendo los controles de flujo de programa con los que usted probablemente tenga ya alguna experiencia, GOTO, GOSUB /RETURN, ON...GOTO y ON...GOSUB/RETURN.

He aquí un programa simple que nos lo muestra en acción:

```
10 PRINT "Este es el QL ";
20 GOTO 10
```

Como habrá imaginado, esto nos llenará la pantalla con lo siguiente, hasta que usted corte el programa:

```
Este es el QL Este es el QL Este es el
QL Este es el QL Esta es el QL Este es
el QL Este es el QL Este es el QL Este
es el QL Este es el QL Este es el QL E
```

En este programa, el QL ejecuta la línea 10, después pasa a la línea 20 donde encuentra la instrucción GOTO 10. Como no hemos puesto condiciones para la transferencia de control de la línea 20 a la 10, ésto ocurrirá siempre, y se le denomina bifurcación incondicional.

GOSUB funciona casi de la misma forma, excepto que trabaja en paralelo con otra palabra - RETURN - que dirige la acción de vuelta a la línea posterior a aquella que contenía la palabra GOSUB, como se puede ver en este pequeño programa:

```
5  CLS:CLS #0
10 PRINT "Esta es la lînea 10"
20 GOSUB 50
30 PRINT \"Esta es la lînea 30"
40 STOP
50 PRINT \"Esta es la subrutina en
50"
60 PAUSE 55
70 PRINT \"Aquî retorno"
80 PAUSE 55
90 RETurn
```

Cuando lo ejecute, podrà ver esto en su pantalla:

```
Esta es la lînea 10

Esta es la subrutina en 50

Aquî retorno

Esta es la lînea 30
```

El programa le indica primero que està en la lînea 10, despuès toma el GOSUB 50 de la lînea 20. La secciòn del programa a la que pasa control, que està comprendida entre la lînea a la que se ha pasado control (la 50 en este caso), y el RETurn (90 en nuestro programa), se llama subrutina.

Una vez en la subrutina, el QL se lo indica mediante el mensaje ESTA ES LA SUBRUTINA. Despuès de una pausa le dice AQUI RETORNO y despuès de otra corta pausa el programa toma el RETurn de la lînea 90, que termina la subrutina y manda la acciòn a la lînea posterior a la que le mandò a ella, èsto es, la lînea 30.

El 'nùmero de lînea de destino' para un GOTO o GOSUB puede ser el resultado de una operaciòn aritmètica (como X=A/B: GOTO X) o de alguna expresiòn (como GOSUB 8*X). Tenga en cuenta que si usa expresiones como estas, no debe renumerar su programa con la funciòn RENUM ya que podrîa confundirle

cuando tome la bifurcaciòn.

El QL dispone de unas variantes de GOTO y GOSUB que no poseîan sus hermanos el ZX80, el ZX81 y el Spectrum, estas son, ON...GOTO y ON...GOSUB.

He aquî un programa que nos muestra la sentencia ON...GOSUB en acciòn:

```
10 REMark ON...GOSUB Demo
20 CLS:CLS #0
30 REPeat bucle
40  X=RND(1 TO 4)
50  ON X GOSUB 80,100,120,140
60  PAUSE 36
70 END REPeat bucle
80 PRINT \"Este era un uno"
90 RETurn
100 PRINT \"Este era un dos"
110 RETurn
120 PRINT \"Este era un tres"
130 RETurn
140 PRINT \"Este era un cuatro"
150 RETurn
```

El QL imprimirà ESTE ERA UN UNO cuando el nùmero aleatorio generado en la lînea 40 sea un 1, ESTE ERA UN DOS cuando sea un 2, y asî sucesivamente. Podrà saber cual de los GOSUB va a tomar el ordenador contando los nùmeros de lînea en la sentencia ON...GOSUB. El primero es 80, que significa que el QL irà a la lînea 80 si X es igual a 1. El segundo es 100, asî que el destino del GOSUB cuando X sea 2 serà la lînea 100 y asî sucesivamente.

ON...GOTO funciona de la misma forma, excepto que, por supuesto, no tiene RETurn para devolver la acciòn a la lînea posterior a la sentencia ON...GOTO.

Redundancia

Las sentencias GOTO, GOSUB, ON...GOTO y ON...GOSUB no son

necesarias en el QL, debido a las ricas estructuras de control con las que viene provisto. Sin embargo, las hemos usado en varios de los programas de este libro para hacer un poco màs fàcil de seguir lo que estamos explicando. Generalmente encontrarà que los programas mas complejos no usan GOTO ni GOSUB, asì que espero que llegue a comprender totalmente las nuevas estructuras de control a medida que nos vayamos introduciendo en ellas. De todas formas los programas cortos contienen algunas veces uno o dos GOTOs, ya que, como le he dicho, asì resulta màs fàcil seguir la acciòn del programa.

La Sentencia IF

La sentencia IF es la pieza clave de la capacidad de decisiòn del QL. En efecto, la diferencia principal entre una simple calculadora y un ordenador es la capacidad de èste ùltimo para decidir (si encuentra que una condiciòn es cierta, entonces toma una bifurcaciòn) y despuès actuar de acuerdo con esa decisiòn.

El QL (en comùn con otros ordenadores) puede comprobar determinadas condiciones y usar el resultado para dirigir el siguiente paso que debe ejecutar el ordenador.

Hay tres versiones de la sentencia IF en el QL. La forma màs simple y con la que usted estarà mas familiarizado, es IF/THEN. IF se ha salido la leche, THEN ponte a llorar. IF tu micro BBC se ha roto, THEN còmprate un ZX80.

```
IF ...prueba... THEN ...haz algo...
```

Si el resultado de la prueba es cierto, entonces el QL realizarà los pasos que siguen al THEN. En esta versiòn el IF y el THEN forman una sola sentencia simple. Aqui puede ver unas cuantas formas posibles de lìneas conteniendo la sentencia IF/THEN:

```
IF NOMBRE$="andrès" THEN PRINT "hola"

IF a+b>c THEN LET d=b
```

```
IF jugador=respuesta THEN GOSUB 1000
```

En la segunda forma el THEN no tiene por que estar dentro de la misma sentencia que el IF. Para esta segunda forma necesita una tercera sentencia, END IF , de forma que puede realizar varios 'THEN cambios'.

Si revisa el pequeño segmento de programa que sigue, lo podrà ver en acciòn:

```
10 IF A$="fin" THEN
20    PRINT "Gracias por jugar"
30    LET mayor_puntuacion=puntos
40    LET cuenta_juego=cuenta_juego+1
50 END IF
```

Como puede ver, todas las sentencias entre el IF/THEN inicial (lìnea 10) y el END IF (lìnea 50) son evaluadas sòlamente si la condiciòn que se està comprobando en la lìnea 10 es cierta.

La tercera versiòn del IF/THEN sirve para dos propòsitos. En la segunda versiòn podìa ejecutar o no los pasos que se encontraban dentro de la secciòn IF/END IF. Aquì se le añade la opciòn ELSE, que permite escoger entre dos acciones alternativas. Si nos sigue comprenderà ràpidamente su aplicaciòn:

```
10 IF puntos > record THEN
20    PRINT "Nueva rècord"
30   ELSE
40    PRINT "No superò el rècord"
50 END IF
```

En este programa puede ver que, IF puntos es mayor que el rècord, THEN el programa imprime el mensaje "Nuevo rècord". Si la condiciòn que està comprobando en la lìnea 10 es falsa, entonces funciona el ELSE y el QL imprime "No superò el rècord".

El Then Opcional

En el SuperBASIC la palabra THEN es opcional, como en la mayoría de los ordenadores (el BBC o el ACORN ATOM). El QL sabrá donde debería estar si lo reemplazamos por dos puntos (:) en la primera versión de la sentencia IF/THEN:

```
IF resp=respuesta:PRINT "Esta bien"
```

En las versiones largas de la sentencia, se puede omitir completamente:

```
10 IF nombre$="Andres"
20   PRINT "Bonito nombre"
30   LET puntos=2*puntos
40 END IF
```

De todas formas, aùnque se pueda eliminar el THEN, yo personalmente prefiero incluirlo, ya que ayuda a comprender màs claramente lo que hace un programa.

Anidados

Se pueden anidar tantas sentencias IF/END IF como se considere necesario (siempre que no llene la memoria del QL en el proceso, por supuesto).

Aquí tenemos un programa que anida dos IF/END IF. Así como en los bucles FOR/NEXT anidados cada NEXT està relacionado con el FOR màs cercano, cada END IF se relaciona con el IF precedente màs cercano:

```
10 IF punt=puntuacion THEN
20   IF nombre$="Andrès" THEN
30     PRINT "Bien hecho, Andrès"
40   ELSE
50     PRINT "Lo conseguiste!"
60   END IF
70 END IF
```

Repetición

Hay dos formas fundamentales de controlar la repetición de secciones de código en el QL. El uso de REPeat/END REPeat y FOR/END FOR, en conjunción con NEXT y EXIT.

La estructura de REPeat/END REPeat es como sigue:

```
REPeat etiqueta
 cosas que quiere hacer
 màs cosas para el QL
END REPeat etiqueta
```

Para FOR/END FOR debemos preparar el programa como sigue:

```
FOR etiqueta = rango de repetición
 cosas que quiere hacer
 màs cosas para el QL
END FOR etiqueta
```

Como dijimos anteriormente, estas sentencias se usan normalmente junto con otras dos, NEXT etiqueta y EXIT etiqueta. Cuando el QL pasa por el NEXT, a menos que el rango del FOR se haya excedido, se va a ejecutar la línea o sentencia posterior a la que contiene el FOR o el REPeat correspondiente.

Si el QL llega a la palabra EXIT, se irá a ejecutar la línea que sigue al END FOR o al END REPeat elegido por la línea EXIT. Si está usando EXIT en un bucle, necesita usar END FOR o END REPeat para terminarlo.

Lo mejor de este tipo de construcciones es que le permiten preparar formas complejas de estructuras FOR/NEXT, DO/WHILE o REPeat/UNTIL, pero con la posibilidad de abandonarlas a la mitad sin causar un fallo del sistema (como sucede en la mayoría de los ordenadores cuando se abandonan prematuramente los bucles FOR/NEXT) y sin salirnos de las reglas básicas de la programación estructurada.

Esta última condición puede parecer irrelevante en este momento, pero a medida que vaya desarrollando su habilidad

con el QL, encontrarà que cada vez va concibiendo màs sus programas en forma estructurada. Una vez que haya formado sus ideas en este sentido descubrirà que ha desarrollado una marcada resistencia a escribir programas que, aunque funcionen, estaràn tan amontonados que no se podrà comprender su funcionamiento fàcilmente.

La sentencia EXIT, usada con REPeat y END REPeat, le permite construir elegantes "bucles sin fin" como el que vemos a continuaciòn. Observe este fragmento de programa:

```
10 REPeat tecla
20  a$=INKEY$
30  IF a$="q" THEN EXIT tecla
40 END REPeat tecla
50 :
60 REMark ...resto del programa...
```

El programa se ejecutarà sin descanso desde la lìnea 10 hasta la lìnea 40 (esto es, desde el REPeat tecla_pulsada hasta el END REPeat tecla_pulsada) hasta que, en la lìnea 30, encuentre la tecla "q" pulsada (esto es, que INKEY$ sea igual a "q"). Una vez que èsto haya sucedido, tomara la EXIT usando el balance de la lìnea 30.

He aquì otro programa que nos aclararà aùn màs el REPeat/END REPeat/EXIT:

```
10 REPeat juego
20  puntos=puntos+resultado
30  IF puntos>record:EXIT juego
40 END REPeat juego
50:
60 REMark ...resto del programa...
```

Seleccionando una Direcciòn

El vocabulario del QL contiene las sentencias SELect y END SELect, que le proporcionan otro mecanismo para determinar què es lo que debe hacer a continuaciòn. El valor de la variable es comparada y la decisiòn se toma de acuerdo con

el valor encontrado.

En contraste con IF/THEN (donde debe realizar una acciòn o saltar) e IF/THEN/ ELSE (donde puede ejecutar una de dos condiciones, o saltar ambas) SELect/END SELect permite al QL elegir entre varias respuestas alternativas.

Verà que tambièn puede añadir una sentencia ON REMAINDER que representa el curso de una acciòn que puede tomar el QL si todas las pruebas son negativas (ninguno de los valores de variable especificado corresponden a una acciòn particular a seguir).

Aquì puede ver la apariencia que puede tomar un programa:

```
100 SELect ON nivel
110  ON nivel=1000
120    PRINT "No lo has hecho bien"
130    bonificacion=-200
140   ON nivel=2000
150     PRINT "Eres Capitàn"
160     bonificacion=780
170   ON nivel=3000
180     PRINT "Eres Comandante"
190     bonificacion=1560
200   ON nivel=REMAINDER
210     PRINT "Eres Emperador"
220     bonificacion=3500
230 END SELect
```

Si ejecuta este programa, podrà ver que imprime "No lo has hecho bien" y pone la variable bonificacion a 200 si nivel es igual a 1000; "Eres Capitàn" con bonificacion a 780 si nivel es 2000 y asì sucesivamente. Si nivel no es igual a 1000, 2000 o 3000, ON REMAINDER entra en acciòn y pone bonificacion en 3500 e imprime el mensaje "Eres Emperador".

No hace falta que incluya toda la informaciòn que ve en las lìneas 110, 140, 170 y 200 del programa anterior. Esta otra versiòn realiza exactamente lo mismo:

```
100 SELect ON nivel
```

```
110 =1000
120     PRINT "No lo has hecho bien"
130     bonificacion=-200
140 =2000
150      PRINT "Eres Capitàn"
160      bonificacion=780
170 =3000
180      PRINT "Eres Comandante"
190      bonificacion=1560
200 =REMAINDER
210      PRINT "Eres Emperador"
220      bonificacion=3500
230 END SELect
```

Reemplazando IF/THEN

También puede usar SELect (sin END SELect) para obtener una sentencia màs clara que con IF/THEN pero que realize la misma función. Para que comprenda lo que quiero decir, aquì tenemos algunas sentencias IF/THEN:

```
10 IF result>500 AND result<1000 THEN
PRINT "Mediocre"
20 IF result>999 AND result<2000 THEN
PRINT "Aceptable"
30 IF result>1999 AND result<3000 THE
N PRINT "Mejoras"
```

Puede usar SELect para producir un programa màs claro, como el que vemos a continuación:

```
10 Select ON result
20 ON result=501 TO 999 THEN
PRINT "Mediocre"
30 ON result=1000 TO 1999 THE
n PRINT "Aceptable"
40 ON result=2000 TO 2999 THE
N PRINT "Estas mejorando"
50 END SELect
```

Aunque este programa no es màs corto que el de la versiòn

IF/THEN, nos muestra màs claramente què valores debe tener la variable result para producir un determinado efecto. En contraste con esto, la versiòn IF/THEN requiere mayor trabajo para comprender su lògica (esto es, debe pensar que <2000 significa 'hasta, pero no igual a 2000' y asì sucesivamente).

Aquì le ofrecemos una versiòn màs corta:

```
10 SELect ON result
20  =501 to 9999 : PRINT "Mediocre"
30  =1000 TO 1999 : PRINT "Aceptable"
40  =2000 TO 2999 : PRINT "Estas mejo
rando"
50 END SELect
```

Capítulo Nueve

Funciones y Procedimientos

Las funciones y los procedimientos son elementos vitales en el lèxico del QL. Una vez definido, usando DEFine FUNction y DEFine PROCedure, los podrà incluir en el programa escribiendo simplemente sus nombres como si se trataran de comandos normales.

Una funciòn actùa en conjunciòn con un nùmero o variable y devuelve un valor, de modo que, cuando incluya su nombre dentro del programa, debe ir seguido por este nùmero o variable. La respuesta producida por esta funciòn se obtiene mediante el uso de la sentencia RETurn. Este no es el mismo RETurn con el que se termina una subrutina, con las funciones debe pensar en esta sentencia como el elemento que nos devuelve una respuesta.
He aquì una funciòn tìpica definida al principio de un programa del QL:

```
10 DEFine FuNcion semisuma(x,y)
20  LOCal punto_cinco
30  punto_cinco=INT((x+y)/2)
40  RETurn punto_cinco
50 END DEFine
```

Si quiere ejecutar esta funciòn durante la ejecuciòn de un programa, debe incluir una lìnea como èsta:

```
100 ...
110 PRINT semisuma(99,12)
120 ...
```

Como puede ver, èsto le permite añadir palabras de manipulaciòn aritmètica al vocabulario del QL (como en el FORTH) de forma que, durante el resto del programa, solo necesita referirse al nombre que le ha dado (semisuma) para poder usarla.

Observe el uso de la palabra LOCAL en la línea 20 de la sección de definición de la función. Esto le asegura que el valor dado a la variable punto_cinco no se confundirá con el uso que haga de la variable en otros puntos del programa. Puede tener, por ejemplo, X, Y y Z como variables de tipo LOCAL que pueden actuar con independencia de X, Y y Z a lo largo del programa (donde deben ser variables 'globales'). Le sugiero que no asigne valores a las variables no-locales dentro de una función o procedimiento, porque puede introducir en sus programas problemas difíciles de localizar.

Procedimientos

Los procedimientos son muy similares a las funciones, excepto que usted puede programar acciones dentro de ellos que no sean puramente matemáticas. Los procedimientos pueden alterar los colores de la pantalla, producir sonidos, desplazar la pantalla hacia arriba, manipular cadenas o cualquier otra cosa. Como regla general, debe asumir que toda acción que se pueda realizar con el SuperBASIC, se puede realizar con un procedimiento.

He aquí un programa simple que nos muestra como se define un procedimiento:

```
10 DEFine PROCedure mult_nomb (nom_cuenta)
20   LOCal yo_mismo
30   FOR yo_mismo = 1 TO nom_cuenta
40     PRINT "...su nombre..."
50     PRINT "----------------"
60     PRINT "****************"
70   END FOR yo_mismo
80 END DEFine
```

Para llamarlo incluya simplemente el nombre del procedimiento en cualquier parte de su programa:

```
100 :
110 mult_nomb (6)
120 :
```

Verà como se imprime su nombre, tantas veces como indique el nùmero que ponga entre parèntesis, a continuaciòn de la llamada al procedimiento. Debe incluir èste nùmero (o una variable) solamente si el procedimiento lo requiere. Si no hay nùmero dentro de la definiciòn del procedimiento (como en el siguiente ejemplo) no necesita incluir un par de parèntesis vacìos.

```
10 DEFine PROCedure truncar
20   LOCal b$
30   b$=a$(1 TO (LEN a$)/2)
40   PRINT b$
50 END DEFine
```

Se le llama asì:

```
100 :
110 a$="su nombre"
120 truncar
130 :
```

Como puede ver, los procedimientos le permiten añadir palabras al vocabulario de programaciòn del QL, tal y como se hace en el lenguaje FORTH. Recuerde que, a diferencia de las funciones, que estàn restringidas a manipulaciones matemàticas, los procedimientos pueden ejecutar cualquier cosa que el QL pueda hacer.

Capítulo Diez

Gráficos

El QL posèe una gran cantidad de palabras para controlar la pantalla, las ventanas, el borde de la pantalla, el color y muchas otras cosas. El vocabulario de pantalla incluye:

AT	BLOCK
BORDER	CLS
CSIZE	CURSOR
FLASH	INK
OVER	PAN
PAPER	PRINT
RECOL	SCROLL
STRIP	UNDER

Todos estos (con la excepciòn de PRINT) van seguidos por nùmeros que le dicen al QL exactamente què es lo que usted quiere hacer. En la mayoría de los casos, el nùmero puede tener uno, dos o tres valores (tales como 0, 1 o 2 para SCROLL). RECOL puede tomar uno entre ocho valores (de cero a siete).

Color

Usted puede controlar tres clases de colores en el QL. La primera es el color de fondo (referido en adelante simplemete como 'color'). La segunda puede ser el color de 'contraste' y la tercera es el color producido por lo que en el lenguaje del QL se conoce como "stipple" (punteado). El color se puede controlar con argumentos simples, dobles o triples.

Control del Color

Aquì vemos como se representan los colores dentro del octeto de color. Los tres bits inferiores se encargan del color principal (color), los tres siguientes del color de contraste (XOR del color principal con el de contraste) y los dos bits superiores determinan la muestra de punteado que se ha escogido:

PUNTEADO		CONTRASTE(XOR)			COLOR		
7	6	5	4	3	2	1	0

Si usted quiere un color sòlido (ej. sin punteado), especifique solamente los tres bits inferiores.

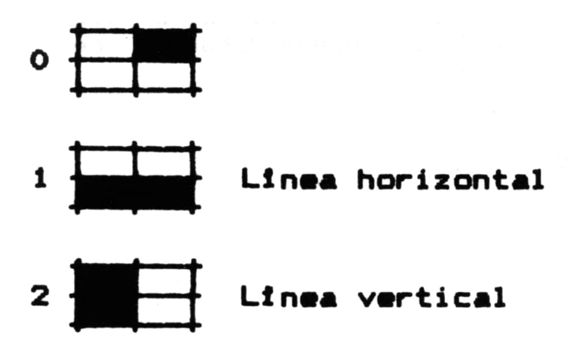
El control de color puede ser simple, doble o triple. Poniendo solamente los tres bits de abajo, para color sòlido, haremos una definiciòn simple (con un solo argumento siguiendo a la sentencia correspondiente, como en PAPER 3). Si selecciona una definiciòn doble, le dice al QL que quiere una definiciòn con contraste (dos nùmeros a continuaciòn de la sentencia, como en Paper 1,5). La definiciòn triple permite seleccionar el color principal, el contraste y el punteado.

Punteados

La reproduciòn del punteado resulta bastante pobre en un televisor. Para verlo adecuadamente, necesitarà un monitor. Sin embargo, como usted descubrirà en su propio televisor, los efectos de punteado pueden ser usados para producir colores diferentes a los no-punteados.

Hay cuatro tipos de punteado, el primero que se ve aquì (marcado como tres) es el punteado que se obtiene si no se le especifica un tipo en particular.





Modos

Como ya hemos dicho anteriormente, hay dos grados de resoluciòn en el QL. El modo elegido (256 o 512) determina no solamente la resoluciòn del dibujo, sino tambièn el nùmero de colores que se pueden obtener. Notarà que hay alguna 'interferencia' entre 'pixels' (puntos o elementos de dibujo) en el modo de alta resoluciòn cuando lo saca por un televisor. Necesita un monitor para verlos correctamente. Cuando se usa la sentencia MODE en un programa, todas las ventanas y canales que corresponden a esas ventanas se "cierran".

MODE 256: En este modo, la pantalla tiene 256 puntos de ancho por 256 de alto y puede usar hasta ocho colores:

0 - negro
1 - azul
2 - rojo
3 - pùrpura o magenta
4 - verde
5 - azul claro o cyan
6 - amarillo
7 - blanco

En un televisor o monitor en blanco y negro obtendremos ocho grados de grises que van desde el negro hasta el blanco.

MODO 512: Este modo pone la pantalla del QL con 512 puntos de ancho y 256. de alto y solo puede disponer de cuatro colores:

```
0 }
  }- negro
1 }

2 }
  }- rojo
3 }

4 }
  }- verde
5 }

6 }
  }- blanco
7 }
```

Aunque este modo reduce el rango de colores que se pueden obtener, el punteado le permite crear una gran cantidad de colores diferentes. He aquí un programa que demuestra el rango de colores disponible en el QL:

```
100 REMark Demostraciòn de colores
110 MODE 256
120 CSIZE 3,1
130 FOR fondo = 7 TO 0 STEP -1
140  FOR contraste = 0 TO 7
150   FOR punteado = 0 TO 3
160 PAPER fondo,contraste,punteado
170 CLS:AT 2,5
180 PRINT "Fondo ";fondo\\;"
Contraste ";contraste\\;"
Punteado ";punteado
190     PAUSE 20
200    END FOR punteado
210   END FOR contraste
220 END FOR fondo
```

Como con cualquier programa, si quiere pararlo antes de

que se haya completado, pulse CTRL y la barra de espaciado.

Print y Palabras Relacionadas

Una de las palabras màs comùnmente usadas en BASIC, es PRINT. Como en otros BASICs, el SuperBASIC usa el comando PRINT para sacar informaciòn a la pantalla. Sin embargo, tambièn permite mandar informaciòn a otros dispositivos, como a la impresora.

La pantalla es el 'dispositivo de salida por omision', que significa que, si no especifica un dispositivo, el PRINT se harà sobre la pantalla. Sin embargo, si ha definido previamente un dispositivo, como una impresora, PRINT mandarà la informaciòn allì, hasta que anule esta instrucciòn del dispositivo con otra.

Su uso màs simple es como sigue:

```
PRINT "Què tal, superestrella?"
```

Esto imprime el mensaje "Què tal, superestrella?" directamente en la pantalla del QL.

La salida de datos del QL incluye el uso de 'separadores', que le permiten dar formato a la salida de PRINT, aunque no estè completamente seguro de la longitud que van a tener los datos individualmente.

Hay cuatro separadores:

! - Este puede ser utilizado como un espacio "inteligente". Pondrà un espacio entre los diferentes datos que imprima en la pantalla o, si la siguiente ha de ser truncada, empezarà una nueva lìnea para ella. Si el espacio cae en el principio de una lìnea, el ! suprimirà el espacio sobrante:

```
100 REMark separador inteligente
110 RESTORE
120 CLS
```

```
130 numero=4
140 DIM a$(4,60),b$(4,60)
150 FOR unid=1 TO numero
160  READ a$(unid),b$(unid)
170  PRINT a$(unid)!b$(unid)
180 END FOR unid
190 DATA"los programas de mi QL","son
 programas de verdad"
200 DATA "mi coche nuevo","solo llega
 a Salamanca"
210 DATA "un poquito de calor","para
el comendador"
220 DATA "haz el amor","y no la guerr
a"
```

, - La coma tabula cada ocho columnas, produciendo columnas claras:

```
100 REMark espaciado con coma
110 CLS
120 total=20
130 FOR cuenta=1 TO total
140  PRINT cuenta,
150 END FOR cuenta
```

\ - La barra inclinada a la izquierda manda los datos a una nueva línea:

```
110 DIM a$(4,20)
120 CLS:RESTORE
130 total=4
140 FOR cuenta=1 TO total
150  READ a$(cuenta)
160  PRINT a$(cuenta)\"       si"
170 END FOR cuenta
180 DATA "comprobando una palabra"
190 DATA "probando un punto"
200 DATA "comiendo una sopa"
210 DATA "apresando un dragòn"
```

; - Finalmente, el punto y coma, asegura que los datos siguen inmediatamente a los anteriores:

```
100 REMark Punto y Coma
110 CLS
120 FOR numeros=33 TO 191
130  PRINT numeros;"-";CHR$(numeros);
"....";
140 END FOR numeros
```

Ejecutando estos programas verà el efecto que crean los diferentes separadores.

INK y PAPER

Como en el Spectrum, INK es el color en el que escribe el QL, PAPER es el fondo sobre el que escribe. Cuando use el comando PAPER deberà usar despuès CLS, para que la pantalla tome el color de fondo que le ha especificado. El formato es simple, debe poner el nùmero que determina el color que usted quiere justo despuès de la palabra PAPER:

```
PAPER 3:CLS
```

Esto pondrà el fondo de color magenta en MODE 256 y rojo en MODE 512.

```
INK 1
```

Esto pondrà azul el color de tinta en MODE 256 y negro en MODE 512.

Una buena combinaciòn de colores para programar, y el que yo suelo usar, es:

```
PAPER 1:INK 7:CLS
```

Que nos darà letras blancas sobre fondo azul.

Limpieza de la Pantalla

Como ya le mencionè anteriormente, debe usar CLS siempre

que ponga un nuevo color de fondo en la pantalla con la sentencia PAPER. Si el CLS no sigue inmediatamente al comando PAPER, simplemente limpia la pantalla de todo lo que hubiera en ella.

Detràs del CLS puede poner un nùmero (del 0 al 4), que indica la parte de la pantalla que debe limpiar:

0 - Limpia la pantalla completa, si no pone ningùn nùmero tomarà el 0 por omisiòn.

1 - Lìmpia toda la parte superior de la pantalla, a ex-eepciòn de la lìnea del cursor.

2 - Casi igual que el 1, excepto que limpia la parte inferior de la pantalla dejando solamente la lìnea del cursor.

3 - Limpia solamente la lìnea del cursor completa.

4 - Limpia solamente la parte de la lìnea que se encuentra a la derecha del cursor, incluido el caràcter sobre el que èste se encuentra.

Si a continuaciòn del CLS pone el sìmbolo (#) seguido de un nùmero, se aplicarà a la ventana que se haya abierto con ese nùmero de canal.

BORDER

El comando BORDER del QL es mucho màs flexible que su omònimo del Spectrum. El BORDER es el àrea que rodea al PAPER. Puede controlar el ancho y el color del borde e incluso poner varios bordes de colores diferentes, unos dentro de otros.

El siguiente programa produce una sucesiòn de colores de BORDER parpadeantes que salen desde el centro de la pantalla, dejando una pequeña àrea de PAPER en el medio:

```
100 FOR w=85 TO 2 STEP -2
```

```
110  BORDER w, RND(0 TO 7)
120  PAUSE 5
130 END FOR w
```

Gràficos Vivos

Por supuesto, se pueden hacer muchas màs cosas con la pantalla ademàs de cambiarle los colores de INK, PAPER y BORDER. Echemos una mirada a algunas de las posibilidades gràficas adicionales del QL.

FLASH: La salida del PRINT puede ser 'fija' (estado normal) o 'parpadeante' (en el que se cambia el PAPER y el INK alternativamente). Este programa nos lo muestra en acciòn:

```
100 REMark FLASH demo
110 PAPER 0:INK 6
120 CLS:CLS #0
130 PRINT \\"  Yo no parpadeo"
140  PAUSE 50TO 100:NEXT j
150 FLASH 10
160 PRINT \\\"  Pero yo si!"
170  PAUSE 70
180 FLASH 0
190 PRINT \\\"  Sin embargo, yo no"
```

Como puede ver, FLASH 1 pone el parpadeo, y FLASH 0 lo quita. El FlASH solo se puede usar con el modo de baja resoluciòn 'MODE 256 o MODE 8).

RECOL: Este comando cambia el color de todo lo que haya en la pantalla sin perder ninguna informaciòn. Funciona cambiando todos los puntos de la pantalla de acuerdo con sus deseos. Esto es, puede usar RECOL para poner todos los puntos que aparecen en verde de color rojo o blanco o como usted elija. RECOL va seguido de ocho nùmeros (pueden estar precedidos por un nùmero de canal).

El primer nùmero despuès del RECOL especifica el

nuevo color para el negro, y los siguientes especifican los nuevos colores para el azul, rojo, magenta, verde, cyan, amarillo y blanco.

He aquí un programa que nos lo muestra en acción:

```
100 REMark RECOL demostración
110 PAPER 0:CLS:CLS #0
120 FOR J=1 TO 740
130  INK RND(0 TO 7)
140  PRINT CHR$(255);
150 END FOR J
160    PAUSE 40
170 BEEP 32000,25
180 RECOL 3,4,5,6,7,0,1,2
190    PAUSE 40
200 BEEP 32000,50
210 RECOL 4,5,6,7,0,1,2,3
220    PAUSE 40
230 BEEP 32000,100
240 FOR J=1 TO 5
250  RECOL RND(0 TO 7),RND(0 TO 7),RN
D(0 TO 7),RND(0 TO 7),RND(0 TO 7),RND
(0 TO 7),RND(0 TO 7),RND(0 TO 7)
260  BEEP 3000,RND(1 TO 10)
270 END FOR J
```

OVER: OVER va seguido de -1, 0 o 1. El número determina la forma en que el QL 'sobreimprimirá', como puede ver en los siguientes programas. Ejecute el primero, que usa OVER -1 en las líneas 160 y 200:

```
100 REMark OVER -1 demo
110 PAPER 0:INK 7
120 CLS:CLS #0
130 AT 3,0
140 PRINT "Veamos el espectáculo en e
l camino"
150 PAUSE 40
160 OVER -1
170 AT 3,0
```

```
180 PRINT "XVeamos el espectàculo en
el camino"
190 PAUSE 40
200 OVER -1
210 PRINT "XVeamos el espectàculo en
el camino"
```

Observe la diferencia cuando el -1 de la lìnea 200 se cambia por 1:

```
200 OVER 1
```

Una vez visto esto, cambie las lìneas 160 y 200 de la siguiente forma:

```
160 OVER 0

200 OVER -1
```

UNDER: Este comando subraya la salida utilizando el color de INK en uso:

```
10 REMark UNDER demo
20 INK 7:PAPER 1:CLS
30 PRINT \"Esto es estàndard"
40 UNDER 1
50 PRINT \\"Esto està subrayado"
60 UNDER 0
70 PRINT \\"Y esto no"
```

UNDER puede ir seguido por un 1 (puesto) o un 0 (quitado).

PAN y SCROLL

Puede usar estas sentencias para producir efectos especiales. En contraste con el SCROLL del ZX81, que mueve la pantalla una lìnea completa, el SCROLL del QL la mueve por puntos, haciendo posible un desplazamiento de la pantalla mucho màs suave. PAN, se puede considerar como un SCROLL lateral, que mueve el contenido de la pantalla a la

izquierda o a la derecha el número de puntos que usted le indique.

PAN: Si pone un número positivo detrás del comando PAN, la ventana que está usando en ese momento, se desplazará hacia la izquierda el número de puntos especificado por dicho número.

Esto significa, que PAN 24, moverá 24 puntos hacia la izquierda todo lo que haya en la pantalla en este momento, rellenando el espacio de la derecha con puntos del color del PAPER en uso. Una instrucción como PAN -65 moverá lo que hay en la pantalla 65 pixels a la derecha, y de nuevo rellenará la izquierda con puntos del color del PAPER que se esté utilizando.

Puede poner hasta dos números detrás de PAN. El primero, como acabamos de ver, controla el movimiento de los puntos (con el signo indicando la dirección). El segundo número puede ser 3 ò 4. Si es 3 (como en PAN -25,3), el QL hará el PAN sólo en la línea del cursor (en este caso 25 puntos a la derecha). Si el segundo número es 4 (como en PAN -25,4) el ordenador hará el PAN en la línea del cursor desde (e incluyendo) la posición de este.

SCROLL: Mueve el contenido de la pantalla hacia arriba o hacia abajo. De forma similar a PAN, puede especificarse la dirección y el número de puntos sobre los que la pantalla hará el SCROLL, así mismo, el espacio que queda se rellena de puntos del color del PAPER que se esté utilizando.

Si el SCROLL va seguido por un solo número, moverá el contenido de la ventana hacia arriba, si el número es positivo (como SCROLL 19 para mover 19 pixels arriba) o hacia abajo si el número es negativo.

Si quiere que el SCROLL sea solamente de una parte de la pantalla, deberá añadir un segundo número.

Añadiendo un 1(como en SCROLL 12,1) le indica al QL que haga SCROLL sòlo de la parte superior de la ventana y un 2 (como en SCROLL 12,2) que haga el SCROLL de la parte baja de la ventana. En ambos casos, la lînea del cursor no està incluida en la parte que se desplaza.

El siguiente programa nos muestra PAN y SCROLL en acciòn. Observe que si parte de la informaciòn desaparece de la pantalla a causa de un PAN o SCROLL, no es restaurada si se vuelve la pantalla a su posiciòn original. Esto lo puede ver en el siguiente programa cuando se pierda el asterisco de la segunda fila a causa de un PAN y la segunda mitad de la segunda fila de letras a causa de un SCROLL:

```
100 REMark PAN demo
110 PAPER 6:INK 2
120 CSIZE 3,1
130 CLS:CLS #0
140 AT 3,3
150 PRINT "Ahî vamos!      *"
160 PRINT "         Ahî vamos!      *"
170 FOR j=1 TO 3
180  PAUSE 20
190  PAN -40
200  PAUSE 20
210  PAN 80
220 END FOR j
230 PAUSE 20
240 PAN -80
250 FOR k=1 TO 3
260  FOR j=1 TO 9
270   SCROLL j
280  END FOR j
290  FOR j=1 TO 5
300   SCROLL -j
310  END FOR j
320 END FOR k
```

Dimensionado

El SuperBASIC del QL, viene equipado con la maravillosa sentencia CSIZE (que sirve para definir el tamaño del caràcter). Esto le permitirà cambiar el tamaño de cada uno de los caracteres que aparazcan en la pantalla, dentro de ciertos lìmites.

Cuando enciende el QL, el tamaño de caràcter se pone a 2,0 si està trabajando en MODE 256, y a 0,0 en MODE 512.

Hay cuatro tamaños de anchura y dos tamaños de altura. Puede verlos en acciòn con este programa:

```
100 REMark CSIZE demo
110 PAPER 0:INK 6:CLS:CLS #0
120 REPEAT ciclo
130 CSIZE 0,0
140 REPEAT bucle
150   INPUT \\"Introduzca altura  (0 a
 1);h
160   IF h=1 OR h=0 THEN EXIT bucle
170 END REPeat bucle
180 REPeat bucle_dos
190   INPUT "Ahora anchura (0 a 3";w
200   IF w>=0 AND w<=3 THEN EXIT bucle
_dos
210 END REPeat bucle_dos
220 CSIZE w,h
230 PRINT \"Asì se ve con "
240 PRINT h;"de alto y ";w;" de ancho
"
250 FOR j=1 TO 1000:END FOR j
260 END REPeat ciclo
```

Como se puede ver en este programa, el ancho puede estar comprendido entre 0 y 3, y el alto entre 0 y 1. Los nùmeros que controlan la anchura de los caracteres son:

CODIGO ANCHO	TAMA\O puntos	MODE 256	MODE 512
0	6	42	85
1	8	32	64
2	12	21	42
3	16	16	32

Y èstos controlan la altura:

CODIGO ALTURA	TAMA\O puntos	FILAS
0	10	25
1	20	12

Trucos de Galàpago

En nuestra explicaciòn de los gràficos del QL debemos incluir una demostraciòn de los gràficos de tortuga. Ya daremos una explicaciòn màs extensa cuando hablemos del programa *QLogo*. Por ahora, introduzca y ejecute el programa que viene a continuaciòn. Es bastante similar a uno que le dimos al principio del libro, asi que si lo salvò en microdrive, le sugiero que lo cargue y vuelva a disfrutarlo de nuevo. Si no lo salvò, aquî tiene el listado:

```
100 REMark Gràficos de tortuga
110 PAPER 0
120 CLS:CLS #0
130 PENDOWN
140 REPEAT logo
150  IF RND>.68 THEN CLS
160  nota=RND
```

```
170  POINT 80,50
180  INK RND(1 TO 7)
190  tortuga=RND(3 TO 70)
200  galapago=RND(25 TO 340)
210   FOR k=1 TO tortuga
220    MOVE k/.7
230    TURN galapago
240    BEEP 32000,k/nota
250   END FOR k
260 END REPeat logo
```

Capítulo Once

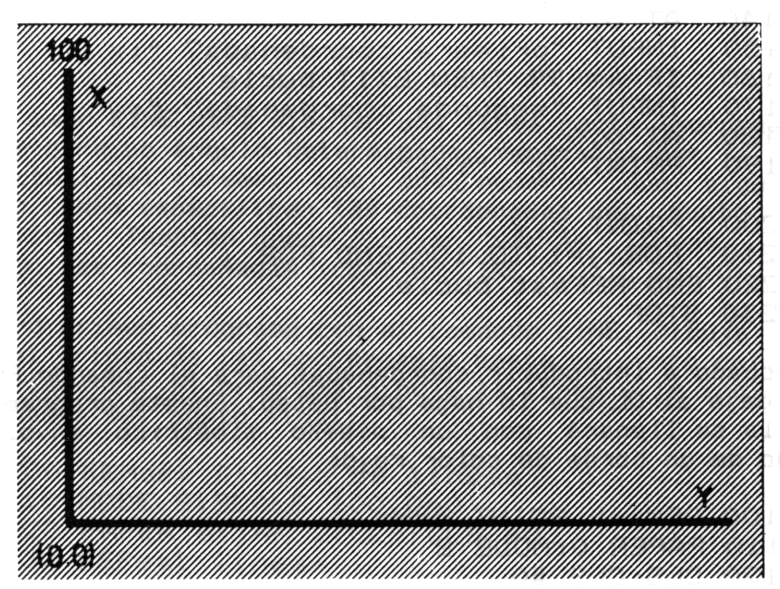
Sistemas de Coordenadas

El Sistema de Gráficos

El 'sistema de coordenadas gráficas' sitúa el origen (0,0) en la parte inferior izquierda de la escala y define la altura de la pantalla como de 100 unidades.

El formato de la ventana determina la longitud del eje X. Sin embargo, la relación con el eje Y se mantiene en uno a uno.

Cinco de los procedimientos gráficos son MOVE (para mover el cursor gráfico), POINT (que marca un punto), ARC (que dibuja un arco de circunferencia), LINE (que dibuja una línea) y CIRCLE (que se usa para dibujar circunferencias o elipses).



Aunque como hemos dicho, una escala de 100 unidades es lo norma, cuando se trabaja en el sistema de coordenadas gráficas, como veremos enseguida, ésta puede modificarse para ajustarla a sus necesidades.

POINT

El comando POINT le permite colocar un punto en la pantalla en una posición que especifica la relación con el origen del gráfico (la esquina inferior izquierda). Generalmente va seguido por dos números, el primero es la coordenada X, y el segundo la coordenada Y.

Pruebe el siguiente programa, introduciendo números a su elección entre 0 y 99. Para terminar introduzca -99 cuando le pida la primera coordenada:

```
100 REMark POINT demo
110 INK 6:PAPER 0
120 CLS:CLS #0
130 REPeat bucle
140  AT 0,0
150  INPUT "primera coordenada? ";x
160  IF x=-99 THEN EXIT bucle
170  INPUT "segunda coordenada? ";y
180  AT 0,0
190  PRINT x;" ";y;"
     "
200  POINT x,y
210  FOR j=1 TO 100:END FOR j
220 END REPeat bucle
230 STOP
```

Ya probado a introducir coordenadas para el comando POINT, ahora deje al QL que lo haga por usted. Eneste programa la pantalla se llenará con un diseño simétrico envolvente rodeado de un borde de color rojo:

```
100 REMark POINT demo
110 PAPER 0:MODE 8
120 CLS:CLS #0
```

```
130 BORDER 2,2
140 REPeat lapso
150  IF INKEY$="" THEN EXIT lapso
160 END REPeat lapso
170 REPeat traza
180  IF INKEY$<>"" THEN EXIT traza
190   IF RND>.5 THEN
200    INK 7
210     ELSE
220    INK 0
230   END IF
240  x=RND(66 TO 166)
250  y=RND(100)
260  POINT x,y
270  POINT 166-x,y
280  POINT 166-x,100-y
290  POINT x,100-y
300 END REPeat traza
```

Puede parar el programa cuando lo desee pulsando una tecla cualquiera.

LINE

Puede usar LINE para dibujar una lìnea recta (sorpresa!) o una que gire un àngulo determinado, empezando por la posiciòn actual del cursor. Los dos nùmeros que siguen al comando LINEn especifican las coordenadas del punto final de la lìnea. Tambièn puede dibujar varias lìneas, usando 'TO' con las coordenadas (como en LINE 0,0 TO 10,10).

Nuestro primer programa con el comando LINE, dibuja lìneas desde el origen de las coordenadas hasta un punto calculado aleatoriamente. El programa dibuja la lìnea y le indica las coordenadas del punto final:

```
100 REMark Dibujo de lìneas
110 MODE 8:SCALE 100,OnO
120 PAPER 6:INK 2:CLS:CLS #0
130 REPeat lineas
140  a=RND(10 TO 100)
```

```
150  b=RND(10 TO 100)
160  AT 0,5
170  PRINT "LINE 0,0 TO ";a;",";b
180  LINE 0,0 TO a,b
190  PAUSE 40
200  CLS
210  IF INKEY$<>"" THEN EXIT lineas
220 END REPeat lineas
```

El siguiente programa nos permite introducir las coordenadas de comienzo (en la lìnea 150), pulsando ENTER entre cada nùmero. Las lìneas que se han introducido anteriormente desaparecen de la pantalla y se dibuja una nueva lìnea segùn sus especificaciones. Tambièn podrà ver las coordenadas elegidas, en la parte superior de la pantalla.

```
100 REMark LINE demo
110 PAPER 0:INK 6
120 CLS:CLS #0
130 REPEAT bucle
140  AT 0,0
150  INPUT "Primeras coordenadas ";x,
y
160  INPUT "Segundas coordenadas ";a,
b
170  AT 0,0
180  PRINT x;",";y;" TO ";a;",";b;"
                                    "
190  LINE x,y TO a,b
200  FOR j=1 TO 2000:END FOR j
210 END REPeat bucle
```

Ahora vamos a producir una versiòn con LINE del ùltimo programa que hemos usado con el comando POINT. Este programa dibuja rectàngulos al azar en la pantalla, muy ràpidamente y en colores tambièn al azar. Si observa detenidamente el listado, verà que lo que hace es unir los puntos creados por el programa del *POINT demo*:

```
100 REMark LINE demo 2
110 PAPER 0:MODE 8
120 CLS:CLS #0
130 REPeat espera
140  IF INKEY$="" THEN EXIT espera
150 END REPeat espera
160 REPeat traza
170  IF INKEY$<>"" THEN EXIT traza
180 INK RND(1 TO 7)
190  x=RND(66 TO 166)
200  y=RND(100)
210  LINE x,y TO 166-x,y
220  LINE 166-x,y TO 166-x,100-y
230  LINE 166-x,100-y TO x,100-y
240  LINE x,100-y TO x,y
250 END REPeat traza
```

Finalmente nos vamos a concentrar en un programa que hemos llamado *Ataque de Estrellas*. Este produce un singular efecto de cristales de colores rotos, que van cambiando constantemente. Puede practicar con esta rutina añadiendo efectos sonoros a los visuales.

```
100 REMark Ataque de Estrellas
110 PAPER 0
120 CLS:CLS #0
130 a=RND(1 TO 70):b=a+30
140 REPeat estrella
150  INK RND(1 TO 7)
160  IF RND>.7 THEN a=RND(1 TO 70):b=
a+30
170  x=65+RND(a TO b):y=RND(a TO b)
180  FOR j=1 TO RND(10 TO 20)
190   LINE x/2,y/2 TO RND(x+15),RND(y
+15)
200  END FOR j
210  IF RND>.6 THEN cls
220 END REPeat estrella
```

CIRCLE y FILL

Veamos ahora otros efectos gràficos de los que contiene el lenguaje del QL. El comando CIRCLE va seguido de tres nùmeros. Los dos primeros especifican la posiciòn del centro del cìrculo y el tercero controla el radio. FILL puede ir seguido por un 1 o un 0, el 0 significa 'no' y el 1 significa 'si'. Ejecute el programa que sigue y entenderà ràpidamente lo efectivo que pueden ser los comandos CIRCLE y FILL combinados:

```
100 REMark Agujeros de bala
110 PAPER 0
120 CLS:CLS #0
130 a=RND(12 TO 74):b=RND(12 TO 88):c
=RND(3 TO 12)
140 REPeat esferas
150  c=c+RND(-2 TO 2)
160  IF RND>.76 THEN a=RND(12 TO 134)
:b=RND(12 TO 88):c=RND(1 TO 15)
170  INK RND(1 TO 7)
180  FILL RND(1 TO 7)
190  CIRCLE a,b,c
200  IF RND>.98 THEN CLS
210 END REPeat esferas
```

Nuestro siguiente programa, *Orbitas*, està basado en el anterior, pero con algunos cambios. El màs significativo està en la lìnea 190 donde, como puede ver hay dos nùmeros adicionales, despuès del tercero que seguìa al comando CIRCLE en el programa anterior. Estos dos nùmeros controlan lo que llamamos 'excentricidad' de la figura dibujada. Un cìrculo tiene una excentricidad de 1, si le variamos este dato produciremos una elipse en la que el àngulo, expresado en radianes, del eje mayor con el de las Xs viene determinado por el segundo de estos nùmeros.

```
100 REMark Orbitas
110 PAPER 0
120 CLS:CLS #0
130 a=RND(12 TO 44):b=RND(12 TO 108):
c=RND(12 TO 88):c=RND(1 TO 8)zd=RND(2
```

```
 TO 4):e=RND(2 TO 4)
140 REPeat ciclos
150  c=c+RND(-2 TO +2)
160 IF RND>.76 THEN a=RND(12 TO 144):
b=RND(12 TO 88):c=RND(1 TO 8):d=RND(2
 TO 4):e=RND(2 TO 4)
170  INK RND(1 TO 7)
180  FILL RND(0 TO 1):IF RND>.4 THEN
FILL 0
190  CIRCLE a,b,c,d,e
200  IF RND>.98 THEN CLS
210 END REPeat ciclos
```

ARC

Desde CIRCLE y FILL pasamos al comando ARC, que dibuja un arco de circunferencia. Así como LINE dibuja una línea recta desde 'x,y TO a,b', ARC dibuja un semicírculo desde el punto 'x,y TO a,b' moviéndose en sentido contrario a las agujas del reloj.

Nuestro programa de demostración nos muestra unos arcos que se van superponiendo y cambiando de color. Los valores del principio y el final aparecen en la parte inferior de la pantalla. La pantalla se limpia frecuentemente. El efecto general es muy llamativo como podrá comprobar cuando lo ejecute.

```
100 REMark ARC en acción
110 PAPER 0
120 CLS:CLS #0
130 REPeat oval
140  INK 6
150  a=RND(3 TO 145):b=RND(1 TO 145)
160  c=RND(1 TO 45):d=RND(1 TO 130)
170  AT 19,5
180  PRINT "a ";a;"   b  ";b;"      c
";c;"   d ";d;"    "
190  FOR j=1 TO 2 STEP .15
200   INK RND(1 TO 7)
210   ARC a,b TO c,d+j,1
220   ARC c,d+j TO a,b,1
```

```
230  END FOR j
240  IF RND>.84 THEN CLS
250 END REPeat
```

Una vez que haya ejecutado *ARC en acción* en su forma original, modifique el programa añadiéndole esta línea:

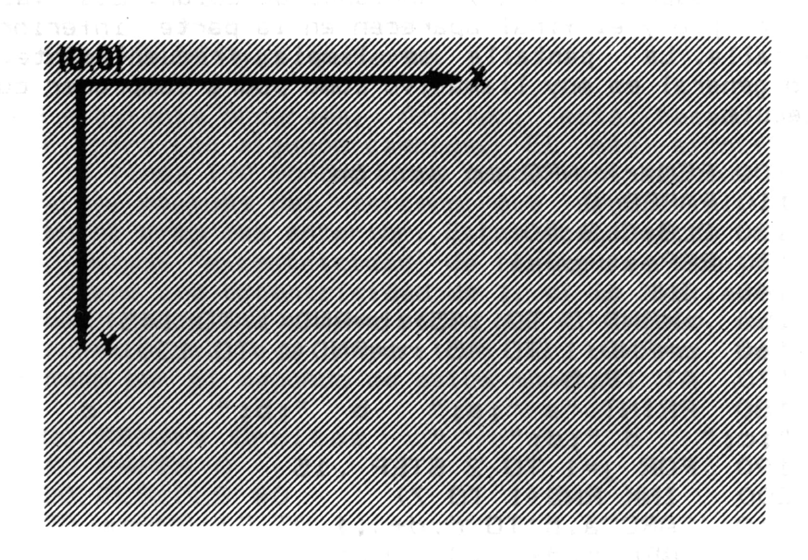
```
115 FILL 1
```

Ahora verá lo rápido que trabaja el comando FILL, borre la línea 115 y añada la siguiente:

```
135  FILL RND(0 TO 1)
```

El Sistema de Puntos

Mientras que el sistema de coordenadas gráficas tiene su origen en la esquina inferior izquierda, el sistema de coordenadas de puntos empieza en la esquina superior izquierda de la pantalla, y se usa para especificar las posiciones y tamaño de las ventanas.



Este sistema, en vez de usar una unidad arbitraria como una centèsima de la altura de la pantalla, se refiere a las posiciones de la pantalla en puntos, asumiendo siempre que el QL està en MODE 512. Sin embargo, no tiene importancia si està en MODE 256 ya que el QL se acomoda automaticamente al modo en el que està trabajando.

Ciertos comandos, como WINDOW, se especifican con relaciòn a la ventana original, mientras que otros, como BLOCK, lo hacen con relaciòn al origen de la ventana en uso.

BLOCK

Este es el primer comando que vamos a estudiar, de los que usan el sistema de puntos. El comando BLOCK va seguido por cinco nùmeros (màs uno opcional que determina el canal en el que debe aparecer). Los dos primeros nùmeros controlan el ancho del bloque en puntos, los dos siguientes determinan su altura y el ùltimo controla el color del bloque.

BLOCK produce algunos efectos verdaderamente ràpidos, como se demuestra en nuestro pròximo programa:

```
100 REMark Ensayo de ls tipos de
                   BLOCK
110 PAPER 0
120 CLS #0
130 REPeat demo
140  CLS
150  z=RND(1 TO 4)
160   SELect ON z
170    =1
180     uno
190    =2
200     dos
210    =3
220     tres
230    =4
240     cuatro
```

```
250    END SELect
260   PAUSE 20
270  END REPeat demo
280  :
290  DEFine PROCedure uno
300   BLOCK 100,75,90,12,2
310   BEEP 4000,103
320  END DEFine uno
330  :
340  DEFine PROCedure dos
350   BLOCK 100,75,240,12,3
360   BEEP 1000,103
370  END DEFine dos
380  :
390  DEFine PROCedure tres
400   BLOCK 100,75,90,115,4
410   BEEP 1000,124
420  END DEFine tres
430  :
440  DEFine PROCedure cuatro
450   BLOCK 100,75,240,115,1
460   BEEP 1900,162
470  END DEFine cuatro
```

Una vez que lo haya ejecutado en esta forma durante un rato, borre la lìnea 260 y verà lo ràpido que puede trabajar este comando.

SCALE

El sistema de puntos permite determinar la escala de la pantalla. Si no especifica ninguna escala, la pantalla tomarà la altura de 100 unidades y el origen estarà en la esquina inferior izquierda. Sin embargon, tambièn se pueden controlar la altura de la pantalla y el origen. El comando que se necesita para ello es SCALE. Los paràmetros por omisiòn de este comando son '100,0,0', donde 100 determina el nùmero de unidades de altura de la pantalla y 0,0 sitùa el origen.

La *Demostraciòn del SCALE* que viene a continuaciòn, dibuja

un 'molino' con el centro de sus aspas en el origen. El programa comienza poniendo SCALE 100,0,0 (usando los valores que se asignan en la lìnea 140). Poniendo el centro de las aspas del molino en el origen, se puede ver exàctamente donde està èste a medida que cambiamos la escala. Cuando ejecute este programa, verà que el origen se mueve gradualmente a travès de la pantalla. El molino se va haciendo màs pequeño a medida que se va dibujando mientras que el valor del SCALE aumenta. En la esquina superior izquierda de su pantalla puede ver los valores de a y b del SCALE:

```
100 REMark Demostraciòn del SCALE
110 PAPER 1:INK 7
120 PENDOWN
130 CLS:CLS #0
140 escala=100:a=0:b=0
150 FOR demo=1 TO 40
160  SCALE escala,a,b
170  CLS
180  AT 1,1:INK 7
190  PRINT "Escala ";escala;"  Origen
";a,b
200  POINT 0,0
210   FOR j=1 TO 9
220   TURN 40
230    INK RND(4 TO 7)
240    FOR k=1 TO 3
250      MOVE 35
260      TURN 120
270     END FOR k
280   END FOR j
290 escala=escala+10:a=a-10:b=b-10
300 BEEP 10000,5*demo:PAUSE 25
310 END FOR demo
```

Capítulo Doce

Gráficos Definidos por el Usuario

En este capítulo veremos un pequeño programa escrito en SuperBASIC, que crea gráficos definidos por el usuario, en el QL. Fuè escrito por Derek Wilson y se publicò por primera vez en la revista *Quanta*.

Antes de nada, he aquí el programa:

```
100 REMark Gráficos definibles
110 REMark    Derek Wilson
120 :
130 RESTORE
140 a=RESPR(64)
150 :
160 FOR i=0 TO 46 STEP 2
170  READ x
180  POKE_W a+i,x
190 END FOR i
200 :
210 DEFine FuNction udg(canal,empie,n
um)
220 LOCal espacio
230 espacio=RESPR(2 + 9*num)
240 CALL a,canal,espacio,empie,num
250 RETurn espacio+2
260 END DEFine
270 :
280 DATA 8302,48,-15620,40,-11839
290 DATA -19986,52,27676,8246,-30720
300 DATA 27926,8256,28672,-27703,9282
310 DATA 5251,21316,5444,1,28709
320 DATA 20035,20085,28922,20085
330 :
340 carac=udg(1,128,16)
```

```
350 :
360 FOR i=0 TO 8
370  POKE(carac + 1),124
380 END FOR i
390 :
400 CLS
410 PRINT CHR$(128)
```

Las sentencias DATA de la lînea 280 en adelante contienen còdigo de màquina, que se introduce en memoria mediante POKE en un àrea reservada por la sentencia RESPR de la lînea 140. Este còdigo de màquina prepara el juego de caracteres requerido, el nùmero de caracteres para los que se ha reservado espacio y el canal para el cuàl se han definido estos caracteres. Asi mismo, verà en el programa una funciòn que nos da la direcciòn inicial donde se pone el modelo en bits, dàndonos una versiòn simplificada de los datos.

El programa se utiliza introducièndolo en el QL y definiendo el juego de caracteres, que nos da una llamada a la funciòn en la lînea 340. Esta, reserva espacio para diez y seis caracteres.

Los caracteres se definen mediante un modelo de bits que debe ocupar solamente los bits 6 al 2 de cada octeto, los bits se cuentan del 0 al 7. Cada caràcter requiere seis octetos consecutivos para definirlo. Por lo tanto, la rutina de la lînea 360 en adelante, prepararà el caràcter 128 como un bloque de un solo color, similar al cursor. CSIZE y las otras sentencias de control funcionan exactamente de la misma forma, con los caracteres definidos por el usuario, que con los caracteres del juego original.

He aquî algunos caracteres que creo que le gustarà probar. Cambie las lîneas desde la 350 para que se lean como sigue, despuès ponga los datos de su caràcter en la lînea 420.

El primero de ellos, convierte el bloque en un 7:

```
350 :
360 FOR i=0 TO 8
```

```
360  READ z
370  POKE(carac + 1),z
380 END FOR i
390 :
400 CLS
410 PRINT \\"  ";CHR$(128);"   ";CHR$
(128):STOP
420 DATA 0,124,4,8,16,32,64,0,0
```

Este es el parèntesis de la derecha:

```
420 DATA 0,48,8,8,8,48,0,0,0
```

Una especie de invasor del espacio:

```
420 DATA 0,124,124,108,40,108,0,0,0
```

Un huso:

```
420 0,16,56,68,56,16,0,0,0
```

Una pequeña torre:

```
420 DATA 0,24,60,36,60,36,60,0,0
```

Y finalmente una bomba o un hombre cayendo (depende de como se mire):

```
420 DATA 0,108,40,16,124,56,16,0,0
```

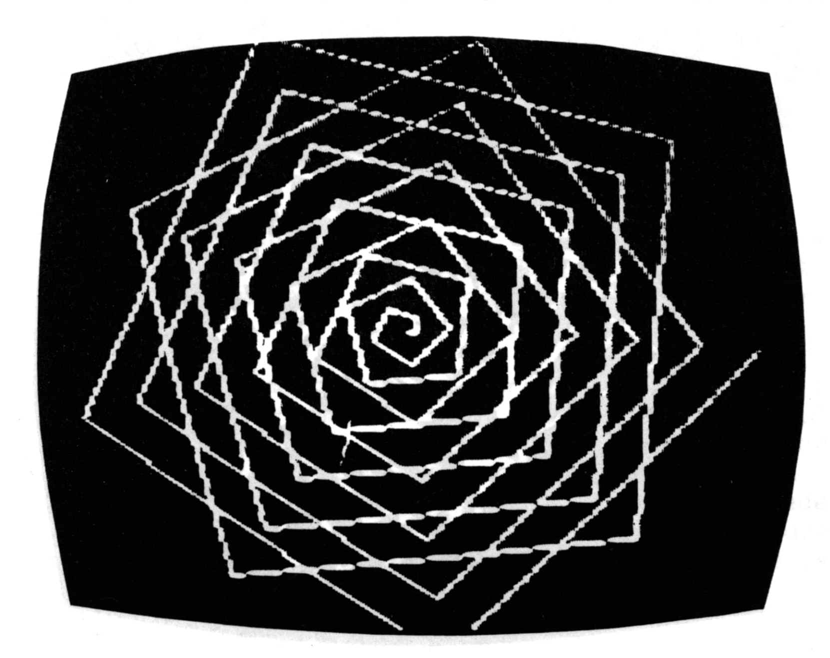
Puede divertirse un buen rato experimentando los gràficos en su QL, con esta rutina.

Capítulo Trece

QLogo

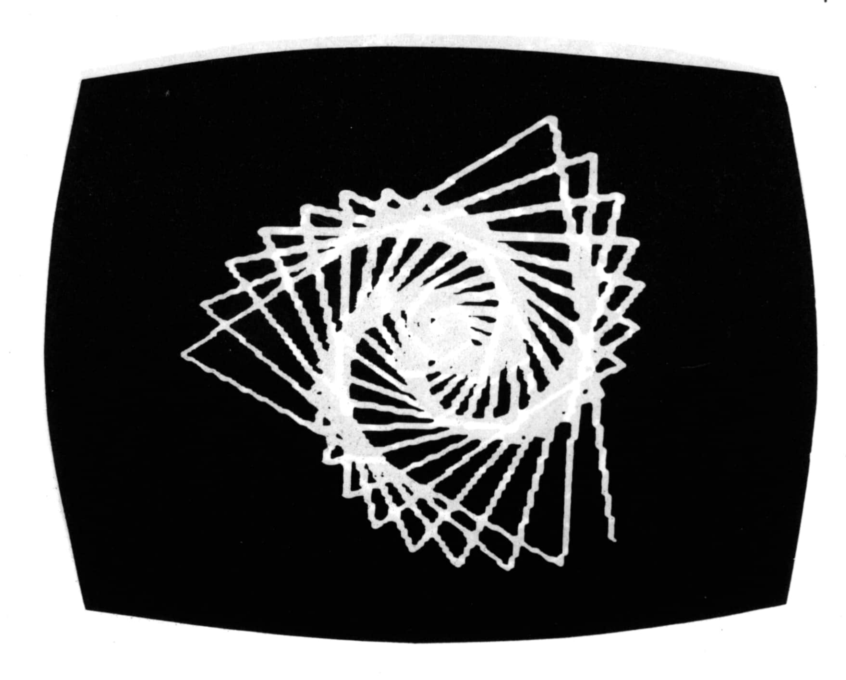
El LOGO es un lenguaje de programaciòn muy diferente del BASIC. El BASIC se creò para que fuera sencillo de aprender, pero no resulta fàcil de ampliar. El LOGO se diseñò como un lenguaje simple y fàcil de aprender pero con posibilidades de ampliaciòn; que sirviera para 'enseñar aprendiendo'. Los creadores de este lenguaje consiguieron cubrir totalmente estos objetivos.

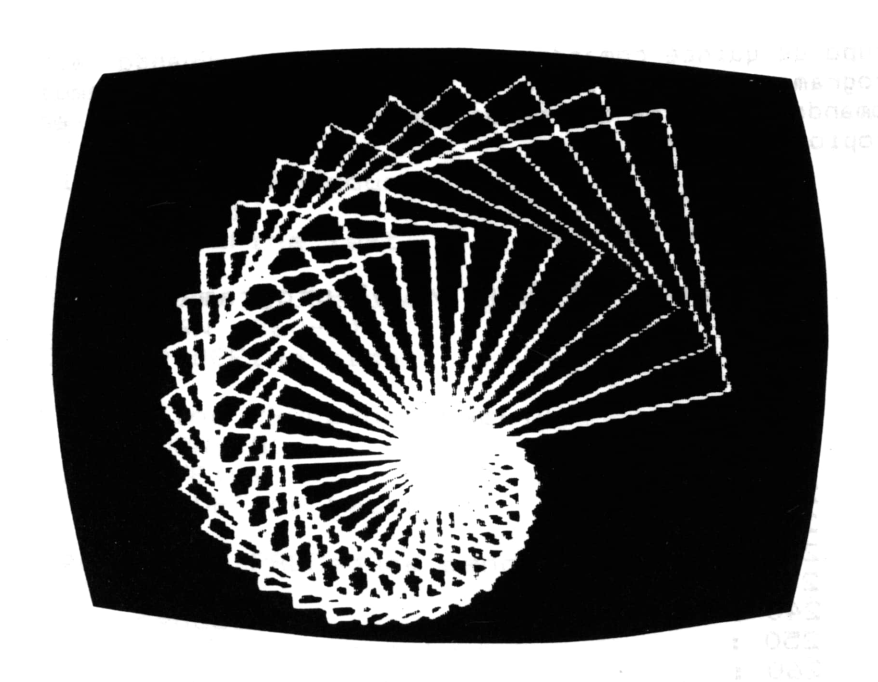
Los *gràficos de tortuga* que contiene el QL pueden ser combinados para producir muchos otros comandos que constituyen el lenguaje ordinario del LOGO. En este capítulo trabajaremos sobre esta idea, produciendo un lenguaje al que hemos llamado *QLogo*.



El LOGO fuè diseñado por el Dr Simon Papert, cuando era profesor de Matemàticas y Educaciòn en el Instituto de Tecnologìa de Massachusetts, en Estados Unidos. Se trasladò posteriormente a Francia para dirigir el World Computer Centre. Papert decìa, que el primer lenguaje de ordenador que aprendìa una persona, marcaba el camino que seguirìa en su carrera y su forma de pensar sobre los programas el resto de su vida. Sostenìa que los comienzos con el LOGO pavimentaban su camino mucho mejor que con el BASIC. Por supuesto que en esa etapa no se habìan diseñado BASICs de la calidad del SuperBASIC (o del BBC BASIC, que tambièn contiene estructuras de control muy refinadas).

Papert aprendiò de las observaciones hechas por Jean Piaget, que mantenìa que los niños son capaces de adquirir habilidades realmente complejas -como hablar y andar- sin un entrenamiento formal. Asì mismo, el entorno informal que les impulsa a aprender està ausente en las aulas normales. Papert se dispuso a crear un lenguaje de ordenador que pudiera remediar esta deficiencia. De acuerdo con Papert,





la mayoría de las instrucciones de programación de ordenadores, ponen al aprendiz casi en la posición de ser programado por el ordenador. El LOGO, por el contrario, pone al niño firmemente en su posición de programador de la máquina.

Esto lo hizo permitiendo al programador crear nuevos modelos y acciones -como una que dibuje un triangulo- y deja al ordenador que lo ejecute simplemente tecleando el comando TRIANGLE. El BASIC ordinario, no tiene esta forma de crear nuevos comandos y funciones. Esto puede ser emulado en SuperBASIC creando *procedimientos* que usen los comandos gráficos de tortuga para crear nuevos modelos.

Los comandos, disponibles en SuperBASIC, para los gráficos de tortuga son: PENUP, PENDOWN, MOVE, TURN y TURN TO. Estos

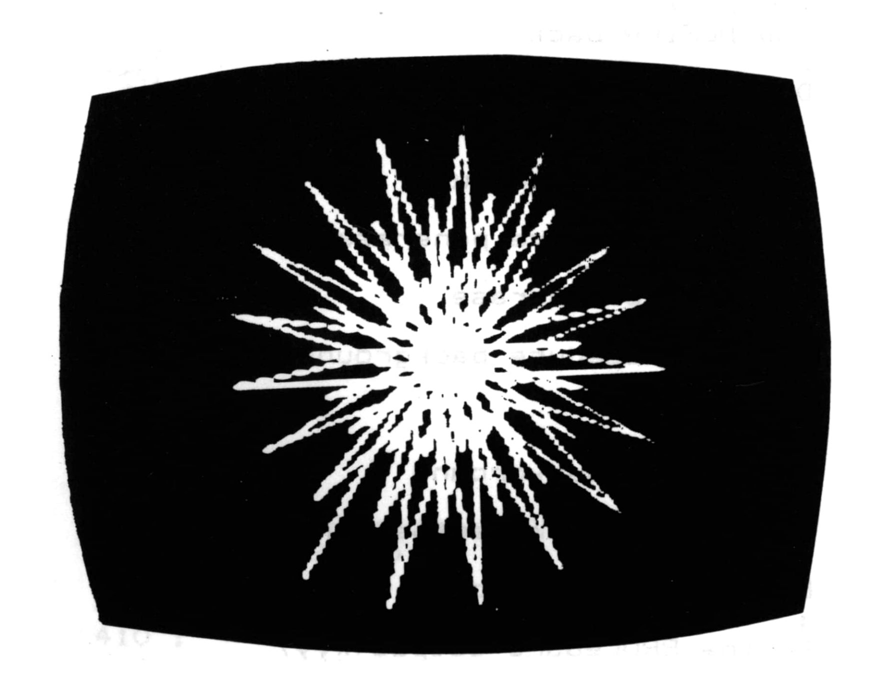
pueden ser combinados con otras opciones del SuperBASIC para emular la mayoría de los comandos del LOGO, como HOME, CLEARSCREEN, SETPOS, SUM y REMAIN.

Introduzca el siguiente programa, para darle al QL un grupo de quince comandos para el *QLogo*. Cuando esté el programa en memoria, vuelva al libro y le explicaremos los comandos de forma que pueda empezar a combinarlos en sus propios programas *QLogo*.

```
100 REMark QLogo
110 inicializar
120 :
130 REMark ** Aquí va el programa **
140 :
150 :
160 :
170 :
180 :
190 :
200 :
210 :
220 :
230 :
240 :
250 :
260 :
270 :
280 :
290 :
300 :
310 :
320 :
330 :
340 :
350 :
360 :
370 DEFine PROcedure HOME
380  POINT 80,50
390  setheading 0
400 END DEFine HOME
410 :
```

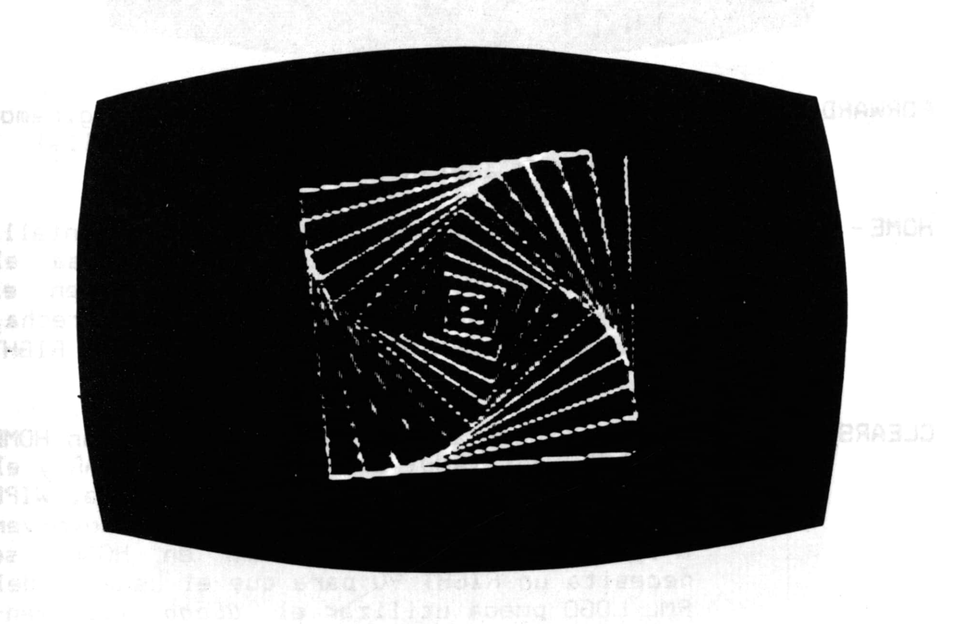
```
420 DEFine PROCedure clearscreen
430  HOME
440  CLS
450 END DEFine clearscreen
460 :
470 DEFine PROCedure forward(x)
480  MOVE x
490 END DEFine forward
500 :
510 DEFine PROCedure left(x)
520  TURN(x)
530 END DEFine left
540 z
550 DEFine PROCedure right(x)
560  TURN (360-x)
570 END DEFine right
580 :
590 DEFine PROCedure back(x)
600  TURN 180
610  MOVE(x)
620 END DEFine back
630 :
640 DEFine PROCedure pencolour(x)
650  INK x
660  pencol=x
670 END DEFine pencolour
680 :
690 DEFine PROCedure penerase
700  INK backgrnd
710 END DEFine penerase
720 :
730 DEFine PROCedure background(x)
740  PAPER x
750  backgrnd=x
760  clearscreen
770 END DEFine background
780 :
790 DEFine PROCedure setheading(x)
800  TURNTO 450-x
810 END DEFine setheading
820 :
830 DEFine PROCedure setpos(x,y)
```

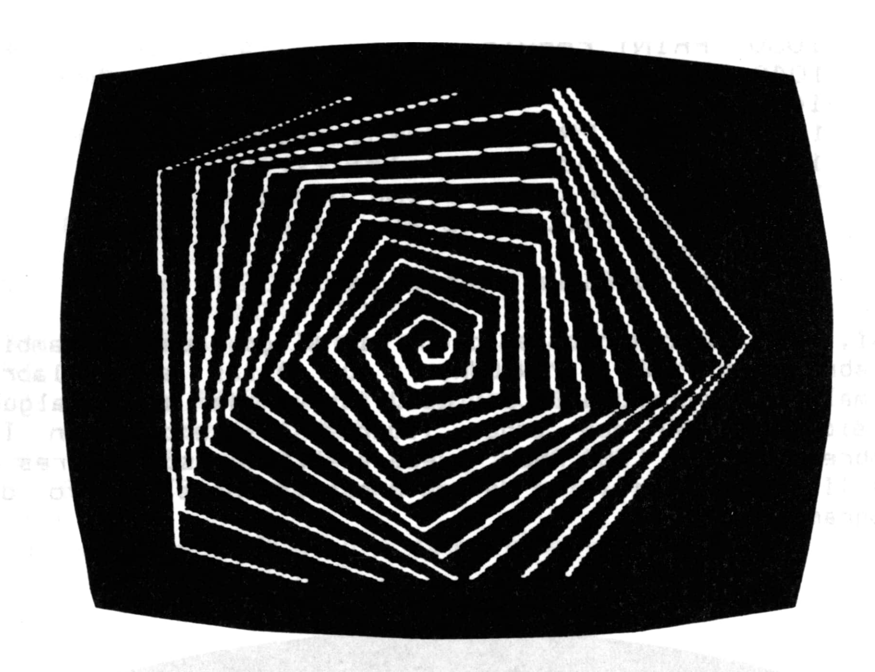
```
840  POINT x,y
850 END DEFine setpos
860 :
870 DEFine PROCedure sum (x,y)
880  result=INT(x+y)
890  PRINT result
900 END DEFine sum
910 :
920 DEFine PROCedure random(x)
930  result=RND(0 TO x-1)
940 END DEFine random
950 :
960 DEFine PROCedure product (x,y)
970  result=INT(x*y)
980  PRINT result
990 END DEFine product
1000 :
1010 DEFine PROCedure remain(x,y)
1020  result= x MOD y
```



```
1030  PRINT result
1040 END DEFine remain
1050 :
1060 DEFine PROCedure initialise
1070  backgrnd=0:pencol=6
1080  PAPER backgrnd:INK pencol
1090  MODE 8:CLS:CLS #0
1100  home
1110 END DEFine initialise
```

Así, como los comandos añadidos al *QLogo*, usarà tambièn palabras como PENUP y PENDOWN, ya que èstas son palabras normales del vocabulario del LOGO. (Si ya ha usado alguna versiòn del LOGO que tenga ligeras diferencias en los nombres, como el LOGO RML., puede sustituir esos nombres en las lìneas de definicion de los procedimientos dentro del programa).





FORWARD X- Mueve la tortuga invisible (que imaginamos que està dibujando una lìnea tras ella) X pasos hacia adelante.

HOME- Pone la trotuga en medio de la pantalla mirando hacia arriba(el LOGO RML usa el comando CENTRE, que pone la tortuga en el centro de la pantalla, mirando a la derecha, de forma que CENTRE serà igual a HOME: RIGHT 90).

CLEARSCREEN- Limpia la pantalla y realiza tambièn un HOME (el LOGO del TANDY usa el comando CLEAR y el RML usa CLEAN para limpiar la pantalla, WIPE para limpiar el texto y CENTRE para devolver a la tortuga al centro. Como en HOME, se necesita un RIGHT 90 para que el usuario del RML LOGO pueda utilizar el *QLogo* fàcilmente).

LEFT X- Gira a la tortuga X grados a la izquierda.

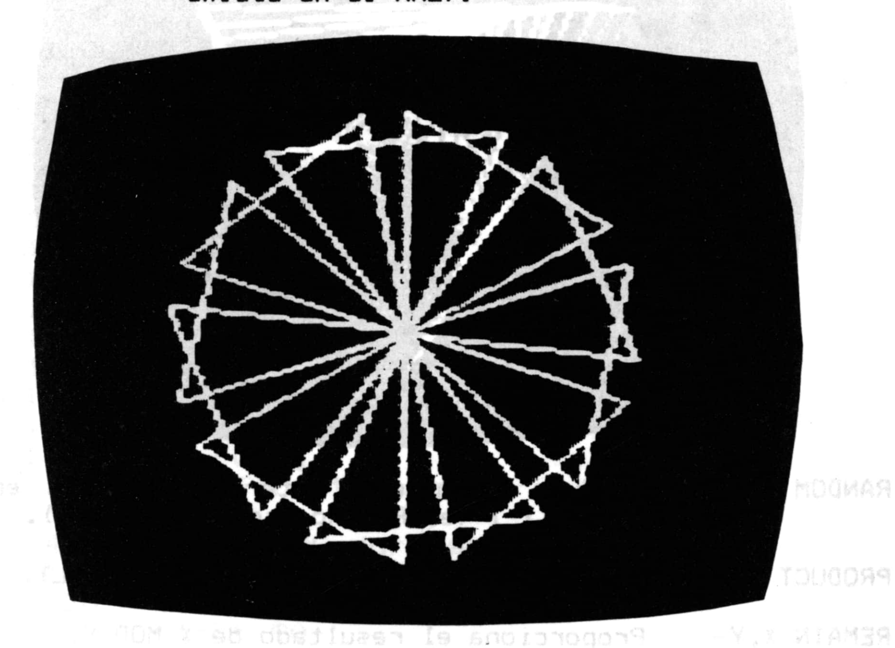
RIGHT X- Gira a la tortuga X grados a la derecha.

BACK X- Permite a la tortuga moverse hacia atràs X pasos (el RML LOGO, lo llama BACKWARD o BK).

PENCOLOUR- selecciona el color, de los normales del QL en MODE 8, con el que dibujarå la tortuga (en RML LOGO, lo llaman PENCIL).

PENERASE- Con este comando, la tortuga dibuja con tinta del color del fondo, con lo que puede borrar la lînea que ha dibujado (el TANDY LOGO tiene un comando equivalente mientras que el RML lo llama ERASER).

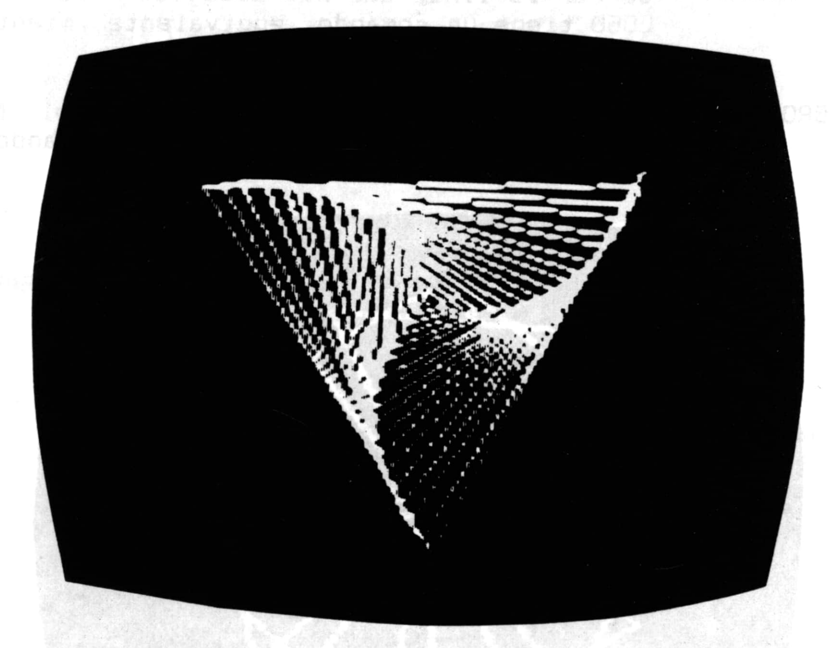
BACKGROUND X- Elige un color de PAPER (de nuevo del MODE 8) y realiza un CLEARSCREEN (este comando no existe en el RML).



SETHEADING X- Coloca la cabeza de la tortuga apuntando en la dirección especificada (cero grados corresponden a la parte superior de la pantalla y 90 a la derecha).

SETPOS X,Y- Mueve la tortuga a la posición especificada por las coordenadas X e Y, poniendo un punto en esa posición. (el TANDY dibuja una línea desde la posición inicial hasta la nueva; ACORNSOFT LOGO llama a este comando SETXY, mientras que en el RML es SETX, SETY).

SUM X,Y- Suma los números X e Y imprimiendo el resultado (el RML no tiene este comando).



RANDOM X- Produce un número entero aleatorio entre cero y X-1 (el RML no produce enteros).

PRODUCT X,Y- Multiplica X e Y (no existe en el RML).

REMAIN X,Y- Proporciona el resultado de X MOD Y.

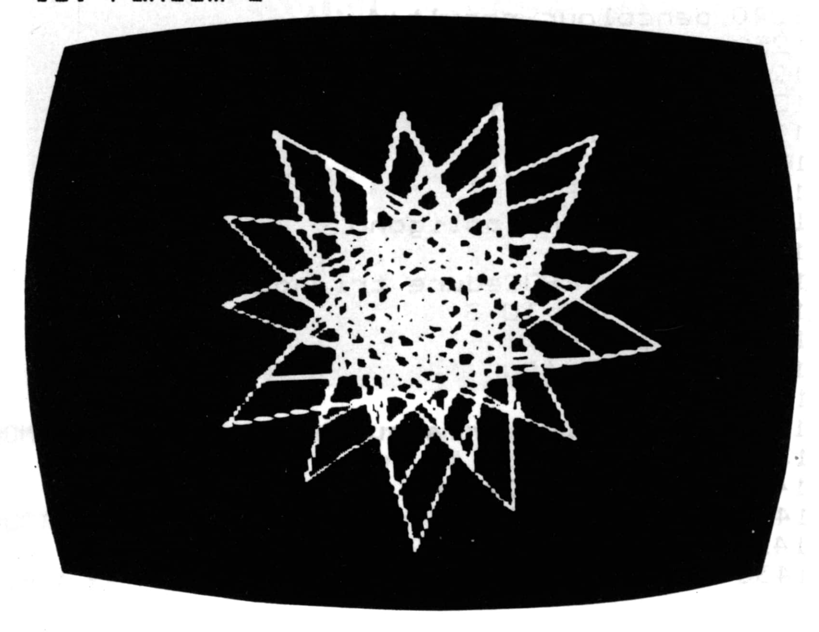
Para usar el *QLogo*, debe introducir su programa en las líneas en blanco, desde la 140 hasta la 360 (puede renumerar el programa si quiere tener más espacio). Puede llamar a los procedimientos definidos para crear sus propios comandos. También puede añadir rutinas preparadas para llamarlas desde su programa. He aquí algunas rutinas preparadas que aparecen al final del listado del *QLogo*, CUADRADO, POLIGONO, TRIANGULO, ESCALERA E HILADO:

```
1120 :
1130 REMark rutinas preparadas
1140 :
1150 DEFine PROCedure cuadrado(x)
1160  FOR j=1 TO 4
1170   forward x
1180   right 90
1190  END FOR j
1200 END DEFine cuadrado
1210 :
1220 DEFine PROCedure poligono(w,x,y,
z)
1230 random 4
1240 pencolour result+4
1250  FOR k=1 TO w
1260   FOR j=1 TO x
1270    forward y
1280    right z
1290   END FOR j
1300  END FOR k
1310 END DEFine poligono
1320 :
1330 DEFine PROCedure triangulo(x)
1340  FOR j=1 TO 3
1350   forward x
1360   right 120
1370  END FOR j
1380 END DEFine triangulo
1390 :
1400 DEFine PROCedure escalera(x,y)
1410  FOR j=1 TO x
1420   forward y
1430   right 90
```

```
1440    forward y
1450    left 90
1460   END FOR j
1470  END DEFine escalera
1480  :
1490  DEFine PROCedure hilado(x,y,z)
1500   POINT x,y
1510   FOR j=1 TO 30
1520    forward z
1530    right 12
1540   END FOR j
1550  END DEFine hilado
1560  :
```

Aquí tiene un programa que puede insertar en las líneas en blanco para mostrar las rutinas preparadas en acción:

```
140 cuadrado 30
150 PAUSE 60
160 clearscreen
170 :
180 random 8
```



```
190 w=result+2
200 random 12
210 x=result+6
220 random 25
230 y=result+6
240 random 50
250 z=result+27
260 poligono w,x,y,z
270 PAUSE 60
280 clearscreen
290 :
300 random 8:x=result+14
310 random 8:y=result+16
310 random 3:z=result+1
330 triangulo (y+z)
340 escalera x,y
350 hilado x,y,z
360 :
```

Ahora veamos què se puede hacer con ellas individualmente.

Primero veremos el efecto de SQUARE:

```
140 :
150 background 2:pencolour 6
160 x=32
170 FOR accion=0 TO 359 STEP 10
180  setheading accion
190  cuadrado x
200 END FOR accion
210 :
```

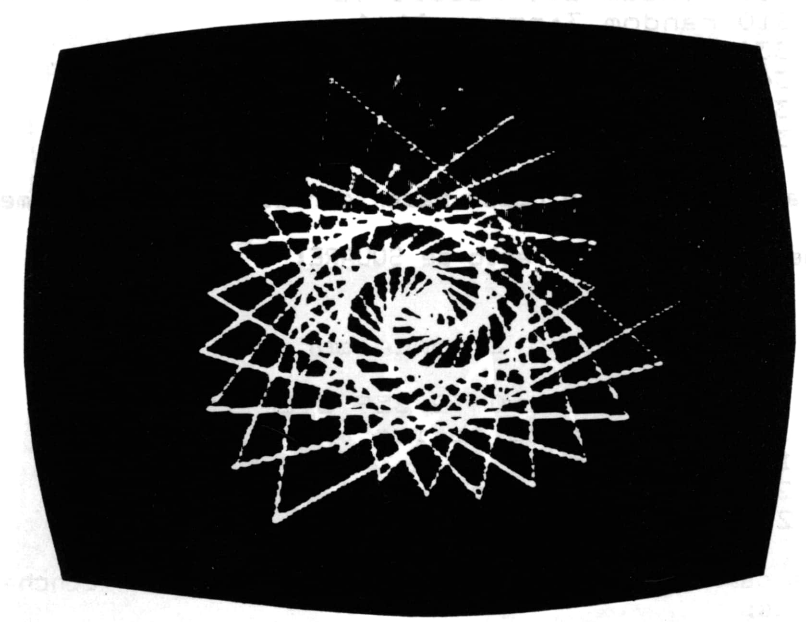
Ahora càmbielo para obtener la figura de una concha de nautilus:

```
140 :
150 background 2:pencolour 6
160 x=1
170 FOR accion=0 TO 359 STEP 10
180  setheading accion
190  cuadrado x + accion/7
```

```
200 END FOR accion
210 :
```

DE la concha a un molino de viento:

```
140 REMark Molino de viento
150 background 0:pencolour 7
160 x=34
170 FOR accion=1 TO 10
180  setheading accion*36
190  triangulo x
200 END FOR accion
210 :
```

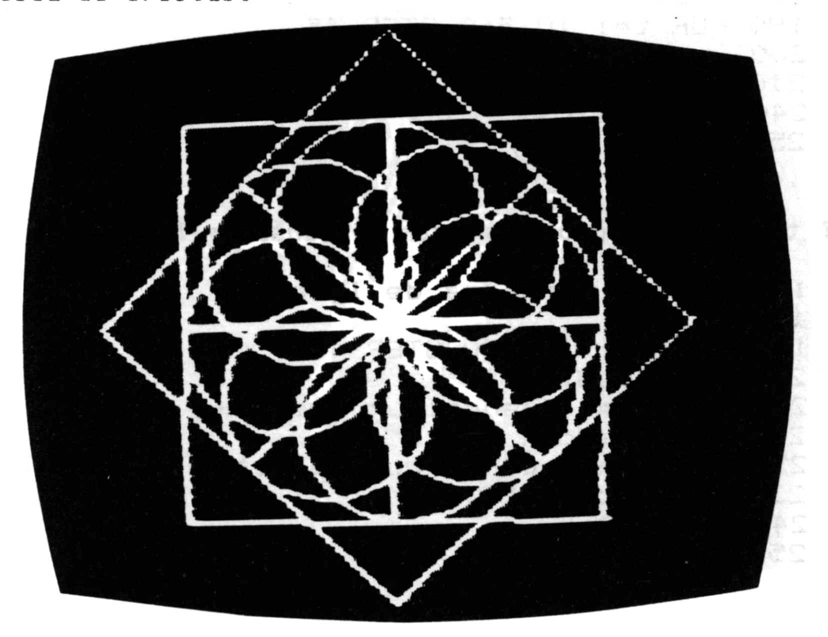


También es posible la animación, como lo demuestra nuestro
Triángulo Giratorio:

```
140 REMark Triángulo giratorio
150 background 0
160 x=24
170 FOR accion=1 TO 20
```

```
180  FOR color=1 TO 2
 190   pencolour 6+color
 200   setheading accion*36
 210   triangulo x
 220  END FOR color
 230 END FOR accion
 240 :
```

Como tributo a las flores de cristal, aquí tenemos una *Magnolia de Cristal*:



```
140 REMark  Magnolia de Cristal
150 background 0
160 :
170 FOR accion=1 TO 100
180  random 4:desp=result
190  IF color=1 THEN x=44:pencolour (
7-desp)
195  IF color=2 THEN z=19:pencolour 5
198  IF color=3 THEN x=30:pencolour (
3+desp)
```

```
200   detheading (accion+desp)*35+5*c
olor
210   triangulo x
220  END FOR color
230 END FOR accion
```

Una *Celosía* es fàcil de dibujar:

```
140 REMark  Celosía
150 x=24
190 FOR t=1 TO 360 STEP 45
200  setheading t
210  cuadrado x
240 END FOR t
250 :
```

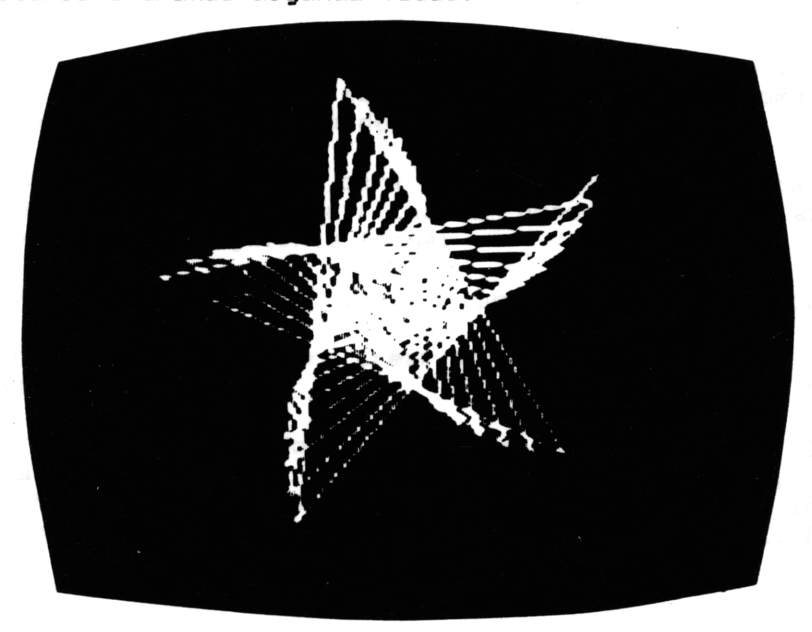
Aquí tenemos dos programas con *Ventanas de Iglesia*:

```
140 REMark  Ventana de Iglesia
150 x=25
190 FOR t=1 TO 720 STEP 45
200  setheading t
205  random 7:pencolour result+1
210  cuadrado x
220  random 7:pencolour result+1
230  hilado 80,50,2.6
240 END FOR t
250 :
```

```
140 REMark Ventana de Iglesia MK II
150 x=32.5
160 pencolour 7
190 FOR t=1 TO 720 STEP 45
200  setheading t
205  FOR g=1 TO 3.5 STEP .5
207   pencolour 3*g
210   square x+g
212  END FOR g
220  pencolour 6
```

```
230  hilado 80,50,3.38
240 END FOR t
250 :
```

Una vez que haya dado RUN al programa *Qlogo* original, puede introducir los comandos directamente y verlos ejecutarse inmediatamente en la pantalla, en vez de incluirlos dentro del programa. Observe que puede usar FILL con el *QLogo*, aunque los efectos que produce pueden resultarle extraños algunas veces.



Si quiere aprender algo más sobre el LOGO y obtener ideas para desarrollarlas en el *QLogo*, los siguientes libros le pueden ser de utilidad:

Logo Programming - Anne Moller(Century,0 7126 0220 8,£6.95)
Discovering Apple Logo - David D. Thornburg(Addison-Wesley, 0 201 07769 8,£10.95)

Capítulo Catorce

Ventanas

Mediante el uso de ventanas se pueden crear varias 'mini-pantallas' dentro de la pantalla real del QL. Cada una de esas ventanas actúa como una pequeña pantalla independiente, que puede realizar todas las funciones de la principal sin interferir en las otras, puede desplazar su texto, modificar los colores de PAPER e INK y usar comandos tales como CLS en cada una de ellas individualmente.

El comando WINDOW se usa para definir las nuevas ventanas. Para usarlo, debe introducir cuatro números (o variables numéricas) de la siguiente forma:

```
WINDOW x_ancho,y_alto,x_principio,y_principio
```

Tenga en cuenta que, para crear ventanas, debe usar las coordenadas de puntos.

Hay un grupo de comandos que se usan en relación con las ventanas, estos aceptan un número extra al final, que determina exactamente como debe proceder el comando. Estos comandos son: BORDER, PAPER, INK, STRIP, PAN y SCROLL. El parámetro extra puede ir desde cero hasta cuatro y tiene el siguiente efecto:

0 toda la pantalla
1 como 0 pero excluida la línea del cursor
2 parte baja de la pantalla excluida la línea del cursor
3 toda la línea del cursor
4 derecha de la línea del cursor, incluido éste

El comando CURSOR se usa para poner el cursor de la pantalla en la posicion que se desee dentro de la ventana.

Si quiere tener más de una ventana en la pantalla al mismo

tiempo, debe designar las ventanas con 'nùmeros de canal'. Este nùmero de canal debe preceder a las coordenadas dentro de la sentencia WINDOW:

```
WINDOW numero_canal,x_ancho...
```

Una vez definida, debe poner el nùmero de canal en todos los comandos que se refieran a esa ventana. Esto es, para modificar el color de INK, por ejemplo, en la ventana 3, debe usar...

```
INK 3,5
```

...donde 3 es el nùmero de la ventana, y 5 es el color de INK.

El siguiente programa, *Ventanas*, crea una ventana en la pantalla siguiendo las instrucciones de la lìnea 170 y "Pulse una tecla...":

```
100 REMark Ventanas
110 PAPER 7:INK 2
120 CLS:CLS #0
130 REPeat bucle
140  IF INKEY$="" THEN EXIT bucle
150 END REPeat bucle
160 AT 0,0
170 PRINT "Pulse una tecla para forma
r una ventana"
180 REPeat bucle
190  IF INKEY$<>"" THEN EXIT bucle
200 END REPeat bucle
210 PAPER 2:INK 6
220 WINDOW 100,100,90,80
230 CLS
240 AT 3,0
250 PRINT "Hola, què tal?"
```

Puede definir varias ventanas en la pantalla al mismo tiempo, como verà en nuestro pròximo programa de demostraciòn. Las ventanas se abren usando una sentencia OPEN (vea las lìneas 140 a 170). El comando OPEN se usa

para iniciar dispositivos, un tèrmino que es demasiado extenso para abarcar solamente las ventanas. El nùmero que sigue el "con_" es el ancho de la ventana. Este es seguido por la altura (los dos van unidos por una "x"), y a continuaciòn (despuès de una "a") las medidas de la ventana a lo ancho y a lo alto (de nuevo con una "x" entre ellos).

Se convencerà de la utilidad de las ventanas con nuestro *Ventanas Demo Mk II*, que abre cuatro ventanas en la pantalla, que se solapan y hace que el texto se mueva dentro de ellas:

```
100 REMark Window Demo MK II
110 PAPER 1:CLS:CLS #0
120 a$="Con veinte cañones por banda,
 viento en popa a toda vela, no corta
 el mar sino vuela un velero berganti
n. Bajel pirata le llaman, por su bra
vura, el temido, en todo el mar conoc
ido, del uno al otro confìn"
130 a$=a$ & a$
140 OPEN #5,con_130x130a90x15
150 OPEN #6,con_130x130a60x60
160 OPEN #7,con_130x130a205x105
170 OPEN #8,con_130x130a180x15
180 REPeat demo
190 PAPER #5,7:CLS #5
200 PRINT #5,A$(1 TO RND(100 TO 418)
210 PAPER #6,6:CLS #6
220 PRINT #6,A$(1 TO RND(100 TO 418)
230 PAPER #7,3:CLS #7
240 PRINT #7,A$(1 TO RND(100 TO 418)
250 PAPER #8,2:CLS #8
260 PRINT #8,A$(1 TO RND(100 TO 418)
270 IF INKEY<>"" THEN EXIT demo
280 END REPeat demo
290 CLOSE #5:CLOSE #6
300 CLOSE #7:CLOSE #8
```

Canales

Usted usarà frecuentemente los canales cuando trabaje con

el QL, de modo que, es importante que tenga una idea clara de lo que son. Muy simple, son las "tuberías" por las que entra y sale la información de los dispositivos que hay conectados al QL.

Antes de que usted pueda usar una de esas tuberías, debe abrirlas con el comando OPEN (y cuando haya terminado con ellas, debe cerrarlas con el comando CLOSE). Algunas de estas tuberías se abren automáticamente cuando enciende el QL. Les llamamos 'canales por omisión', y actúan como 'dispositivos por omisión'. La pantalla es un dispositivo por omisión. Si no especifica, por ejemplo, que la salida del PRINT debe ir a algún otro sitio determinado, se irà a la pantalla. Los canales por omisión no se pueden cerrar.

Los canales se identifican mediante números. Una vez que se ha asignado un número a un canal, este número es todo lo que necesita el QL saber para dirigirse al dispositivo adecuado.

He aquí un sumario de las palabras de control de E/S que usa el QL:

```
BAUD n
CLOSE #n
COPY_N nombre TO nombre
DELETE nombre
DIR [#n,] nombre
FORMAT [#n,] nombre
NET numero de estación de la red
OPEN #n, nombre
OPEN_IN ..
OPEN_NEW ..
```

Y éstas son para dispositivos E/S:

```
SCR[_tamaño][Aposición]
CON[_tamaño][Aposición][_longitud tecla]
SER[n][O/E/M/S][H/I][R/Z/C]
MDVn_nombre
NET[I/O][_otra estación]
```

Capítulo Quince

El Sonido y la Música

En el QL, el sonido se controla mediante el comando BEEP, como en el Spectrum. Sin embargo, a diferencia del Spectrum que aceptaba solamente dos parámetros (que controlaban la duración y el tono), el del QL permite un mayor control, con ocho parámetros con los que jugar.

El comando se usa como sigue:

```
BEEP d,p1,p2,gx,gy,w,f,r
```

En este comando, d es la duración (-32768 a 32767); p1 y p2 son los tonos (0 a 255); gx controla el intervalo de tiempo en que pasa de p1 a p2; gy define la duración de los intervalos p1 y p2;w es el número de veces que se repite el comando automáticamente; f (0 a 15) y r, que añade ruido aleatorio.

Empezaremos nuestra investigación del comando BEEP introduciendo unos programas que usan solamente dos números después del comando. Intente estos dos por ejemplo:

```
10 REMark Brrh!
20 FOR ruido=0 TO 130 STEP 2
30   BEEP 100,ruido
40 END FOR ruido

10 REMark Fluupp!
20 FOR ruido=125 TO 0 step -5
30   BEEP 100,ruido
40 END FOR ruido
```

Como puede oir, los diferentes tamaños de paso en los bucles FOR/END FOR y sus diferentes direcciones, producen sonidos marcadamente diferentes.
Si incluimos un PAUSE dentro del bucle también se producen sonidos diferentes:

```
10 REMark Nnaahh...
20 FOR cont=1 TO 3
30  FOR ruido=85 TO 20 STEP -.5
40   BEEP 5000,225-ruido
50   PAUSE .3
60  END FOR ruido
70 END FOR cont
```

El PAUSE se puede cambiar durante el bucle:

```
10 REMark Torre de control depegando
20  FOR ruido=1 TO 241 STEP -4
30   BEEP 3000,ruido
40   PAUSE 29/ruido
50  END FOR ruido
```

Dentro de cada bucle se puede alojar más de un BEEP para modificar el sonido posteriormente:

```
10 REMark Ataque con laser
20 FOR cont=1 TO 6
30  FOR ruido=4 TO 24 STEP 2
40   BEEP 3000,ruido
50   BEEP 3000,ruido+3
60   BEEP 3000,ruido-3
70  END FOR ruido
80   PAUSE 5
90 END FOR cont
```

Ahora intentaremos incluir ocho números después del BEEP, para ver sus efectos:

```
10 REMark Conexiòn internacional
20  FOR ruido=1 TO 12
30   a=RND(1 TO 10)
40   b=a+RND(1 TO 10)
50   BEEP 10000,a,b,1,2,-9,0,0
60   PAUSE (16+a)
70  END FOR ruido
```

Nuestra siguiente rutina introduce (en la lìnea 70) la palabra BEEPING. BEEPING devuelve cero si el QL no está

produciendo ningùn sonido con el comando BEEP, y un 1 en caso contrario. El bucle de espera REPeat/END REPeat, suspende la ejecuciòn hasta que el sonido haya terminado, antes de permitir que el bucle FOR/END FOR pueda continuar:

```
10 REMark Trasmisiòn de datos
20 FOR sonido=1 TO 9
30  a=RND(1 TO 15)
40  b=15-a
50  BEEP 20000,a,b,1,2,-9,1,1
60   REPeat espera
70     IF BEEPING<>1 THEN EXIT espera
80   END REPeat espera
90 END FOR sonido
```

Se pueden producir sonidos bastante interesantes, demostrando que el comentario del manual que dice que: "es mejor usado experimentalmente que sintacticamente" (que se puede traducir como: "es dificil predecir como va a sonar estudiando solamente los nùmeros"):

```
10 REMark Pequeña batalla
20 FOR sonido=1 TO 11
30  a=RND(1 TO 5)
40  b=105-a
50  BEEP 9000,a,b,1,2,-9,RND(10 TO 15
),1
60   REPeat espera
70     IF BEEPING<>1 THEN EXIT espera
80   END REPeat espera
90   FOR silencio=1 TO 24
100   END FOR silencio
110 END FOR sonido
```

He aquì algunos efectos màs que le gustaràn para usarlos en sus programas:

```
10 REMark Ametralladora
20 REPeat sonido
30 BEEP 32767,2,44,7,1,15,11
40 IF BEEPING THEN GO TO 40
50 END REPeat sonido
```

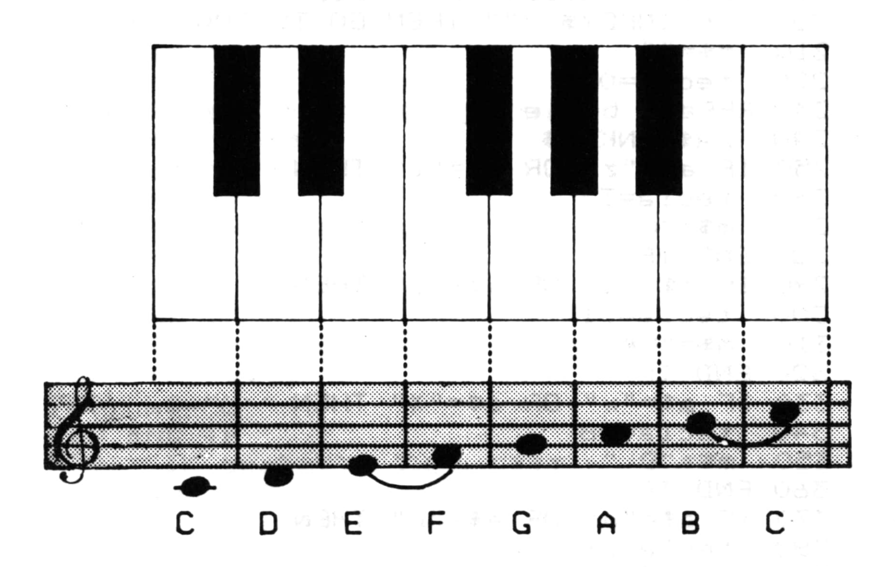
```
10 REMark Blip blip blip-blip blip
20 REPeat sonido
30 BEEP 32767,188,16,7,5,13,0
40 IF BEEPING THEN GO TO 40
50 END REPeat sonido
```

Sería muy interesante una secuencia de efectos sonoros, pero he preferido escribir una rutina simple para hacer generar al QL efectos aleatorios, que le dice los argumentos que está usando para producirlos, de forma que pueda tomar nota de aquellos que le gusten para poder usarlos posteriormente. El QL producirá el sonido y después imprimirá los parámetros en la pantalla. Pulsando cualquier tecla pasará al siguiente sonido:

```
10 REMark Sonidos
20 INK 7:PAPER 1
30 CSIZE 3,0
40 CLS:CLS #0
50 DURACION=32000
60 tono=RND(0 TO 255)
70 tono2=(RND(0 TO 255)
80 gradx=RND(0 TO 15)
90 grady=RND(-8 TO 7)
100 envolv=RND(0 TO 15)
110 vibra=RND(0 TO 15)
120 alea=RND(-32767 to 32767)
130 PRINT \\\"  Tono = ";tono
140 PRINT \"  Tono2 = ";tono2
150 PRINT \"  Gradx = ";gradx
160 PRINT \"  Grady = ";grady
170 PRINT \"  Envolvente = ";envolv
180 PRINT \"  Vibrato = ";vibra
190 PRINT \"  Aleatorio = ";alea
200 BEEP duracion,tono,tono2,gradx,gr
ady,envolv,vibra,alea
210 REPeat colgado
220  IF INKEY$<>"" THEN EXIT colgado
230 END REPeat colgado
240 RUN
```

Nuestro siguiente programa le permite usar el QL como si

fuera un òrgano. En mùsica, tenemos doce semitonos dentro de cada octava, y nuestro òrgano cubrirà una octava, asì que necesitaremos doce notas. La forma màs sencilla de entenderlo es mirando el teclado del piano. Si no tiene un piano a mano, mire el pròximo diagrama, que nos muestra una octava de un piano con los nombres de las notas debajo de cada tecla (tambièn se muestra còmo son las notas escritas).



Aquì està el listado de nuestro programa del òrgano, *El Hombre Orquesta*:

```
10 REMark Hombre Orquesta
20 BORDER 42,0
30 CLS:CLS #0
40 CSIZE 3,1
50 PRINT \"  Musica"
60 anterior=-99
70 REPeat musica
80   teclea
90 AT 3,8:INK RND(4 TO 7):PRINT m$
100 REPeat tocar
110 BEEP 5000,tecla
```

```
120 IF tecla<>anterior THEN
130  EXIT tocar
140  ELSE anterior=tecla
150 END IF
160  END REPeat tocar
170 END REPEAT musica
180 REMark ----------------
190 DEFine PROCedure teclea
200  IF INKEY$<>"" THEN GO TO 200
210  m$=""
220  tecla=0
230 REPeat bucle
240  . a$=INKEY$
250 IF a$="z" OR a$="Z" THEN
260  tecla=33
270  m$="C "
280 END IF
290 IF a$="s" OR a$="S" THEN
300  tecla=31
310  m$="C#"
320 END IF
330 IF a$="x" OR a$="X" THEN
340  tecla=28
350  m$="D "
360 END IF
370 IF a$="d" OR a$="D" THEN
380  tecla=26
390  m$="D#"
400 END IF
410 IF a$="c" OR a$="C" THEN
420  tecla=24
430  m$="E "
440 END IF
450 IF a$="v" OR a$="V" THEN
460  tecla=22
470  m$="F "
480 END IF
490 IF a$="g" OR a$="G" THEN
500  tecla=20
510  m$="F#"
520 END IF
530 IF a$="b" OR a$="B" THEN
```

```
540  tecla=18
550  m$="G "
560 END IF
570 IF a$="h" OR a$="H" THEN
580  tecla=17
590  m$="G#"
600 END IF
610 IF a$="n" OR a$="N" THEN
620  tecla=15
630  m$="A "
640 END IF
650 IF a$="j" OR a$="J" THEN
660  tecla=14
670  m$="A#"
680 END IF
690 IF a$="m" OR a$="M" THEN
700  tecla=12
710  m$="B "
720 END IF
730 IF a$="," OR a$="<" THEN
740  tecla=11
750  m$="C'"
760 END IF
770   IK tecla<>0 THEN EXIT bucle
780  END REPeat bucle
790 END DEFine teclea
```

La fila inferior de teclas del QL (de la Z a la coma) son las teclas blancas, con parte de las teclas de encima de èstas, representamos las teclas negras. He aquì una lista completa de estas teclas, con la tecla del QL en primer lugar, seguida de su equivalente musical entre parèntesis:

Z(C); X(D); C(E); V(F)
B(G); N(A); M(B); ,(C')

S(C#/Db); D(D#/Eb)
G(F#/Gb); H(G#/Ab)
J(A#/Bb)

Las notas no son exactas, pero son suficientemente

aproximadas para simular el sonido de un 'òrgano'. Si quiere afinar màs la exactitud del òrgano, puede intentar experimentar estudiando sintàcticamente la generaciòn de los sonidos.

Capítulo Diez y Seis

Tratando con DATA

READ y DATA son un par de palabras que aparecen en el mismo programa y que están intimamente relacionadas entre sí, aunque aparezcan en sitios muy alejados dentro de un mismo programa.

READ le permite introducirse en un banco de datos creado con DATA, para usarlo dentro del programa. Por ejemplo, el siguiente programa rellena los elementos de la matriz 'brillo$', con una serie de colores:

```
10 DIM brillo$(6,8)
11 RESTORE
12 CLS:CLS #0
20 FOR color=1 TO 6
30   READ brillo$(color)
40 END FOR color
50 DATA "AMARILLO","VERDE ","NEGRO"
60 DATA "PURPURA","AZUL","NARANJA"
```

Para comprobar que los DATA (los elementos de las líneas 50 y 60) han sido introducidos dentro de la matriz por la sentencia READ de la línea 30, puede añadir las siguientes líneas a su programa y ejecutarlo de nuevo:

```
70 FOR color=1 TO 6
80   PRINT brillo$(color)
90 END FOR color
```

Como puede comprobar, el QL ignora las líneas 50 y 60 cuando llega a ellas. Las líneas DATA son leídas solamente cuando se necesitan, pero en cualquier otro momento son ignoradas por el ordenador.

No tiene importancia el emplazamiento de las sentencias DATA dentro de su programa, como podrà ver cuando ejecute

el siguiente programa en su fiel QL:

```
100 DATA "Ruston"
110 REMark DATA demo
120 CLS:CLS #0
130 RESTORE
140 DIM autor$(7,11)
150 FOR libro=1 TO 7
160   DATA"Hartnell","Big Dave"
170 READ autor$(libro)
180   DATA "Gifford","Shaw"
190 END FOR libro
200 FOR salida=7 TO 1 STEP -1
210   PRINT autor$(salida)
220   DATA "Vincent","Hutt"
230 END FOR salida
```

El programa encontrarà fàcilmente los DATA a pesar de estàn esparcidos dentro de èl (alguno de ellos està incluso dentro de un bucle FOR/END FOR), e imprimirà lo siguiente cuando lo ejecute:

```
Hutt
Vincent
Shaw
Gifford
Big Dave
Hartnell
Ruston
```

RESTORE

La tercera palabra que se usa en conjunciòn con READ y DATA es RESTORE. RESTORE mueve el 'apuntador de datos' al principio de un bloque de DATA, como muestra este programa:

```
100 REMark RESTORE demo
110 CLS:CLS #0
120 RESTORE
130 DIM miro$(9,7)
140 FOR veo=1 TO 9
150   IF veo=4 THEN RESTORE
```

```
 160   READ miro$(veo)
 170   PRINT miro$(veo);" ";
 180 END FOR veo
 190 DATA "èsto","es","una","prueba"
 200 DATA "del","uso","del","RESTORE"
```

En un examen ràpido de este programa, usted puede pensar que la salida deberìa ser asì:

èsto es una prueba del uso del RESTORE

Sin embargo, cuando lo ejecute en su QL, verà que imprime esto:

èsto es una èsto es una prueba del uso

El RESTORE de la lìnea 30 mueve el apuntador de datos al principio del DATA cuando 'veo' es igual a cuatro, de forma que la sentencia READ vuelve a empezar de nuevo.

El RESTORE puede ir tambièn a una lìnea especìfica, como hace este programa con la lìnea 150:

```
110 RESTORE
120 CLS:CLS #0
130 :
140 FOR comilon=1 TO 9
150 IF RND(1 TO 8)=4 THEN RESTORE 240
160  READ pizza$
170  PRINT pizza$
180 END FOR comilon
190 DATA "Vegetariana","Margarita"
200 DATA "Cuatro Estaciones","Anchoas"
220 DATA "Napolitana","Milanesa"
230 DATA "Tarta de queso"
240 DATA "Siciliana","Fruta di Mare",
"Maravillosa"
250 DATA "Malo","Derrochòn","Manirroto"
260 DATA "Es el Sr Mostaza"
270 DATA "Pìldora Amarga"
```

Si el nùmero aleatorio generado en la lìnea 150 es igual a

4, entonces el apuntador de datos se va a la línea 240 y comienza a leer por ahí. Esto significa que la salida puede ser (si el 4 es generado despuès del quinto dato leido):

Vegetariano
Margarita
Cuatro Estaciones
Anchoas
Napolitana
Siciliana
Fruta di mare
Maravillosa

DATA se usa normalmente para introducir valores específicos dentro de las matrices al principio de un programa.

Capítulo Diez y Siete

Extendiendo su Vocabulario

Números Aleatorios

Los números aleatorios se usan mucho en juegos y programas de simulación. El QL usa un mètodo particular para especificar los números que usted quiere. La primera forma del comando usa solamente la palabra RND, sin ningún paràmetro:

```
10 REPeat ale_demo
20  PRINT RND;" ";
30 END REPeat ale_demo
```

Si ejecuta este programa en su QL, obtendrà aparentemente una lista de números entre cero y uno.

```
.4514049 .2497407 .3029405 .5438805 7
.759424E-2 .5459563 .8145022 .1482666
 .6346341 .2951227 .7685726 .8432614
```

Si quiere generar enteros aleatorios entre cero y un lìmite superior, debe usar una lìnea como la 30 del siguiente programa:

```
10 limite_sup=7
20 REPeat alea_demo
30  PRINT RND(limite_sup);" ";
40 END REPeat alea_demo
```

Si define el lìmite como 7, obtendrà algo parecido a èsto:

```
7 1 1 6 5 7 5 0 2 1 1 7 1 0 2 4 4 1 5
 5 5 0 3 4 7 5 1 0 2 6 6 4 0 0 5 1 7
0 2 7 3 6 5 4 4 5 4 6 0 2 0 4 6 0 4 4
```

Lo màs probable es que usted quiera, en algún momento,

generar números aleatorios en un rango que no sea desde el cero a un límite superior. En este caso, debe hacer uso de la palabra TO como sigue:

```
PRINT RND(lim_inf TO lim_sup)
```

Dentro de un programa puede aparecer así (que nos dará números aleatorios entre 95 y 105):

```
10 lim_sup=105
20 lim_inf=95
30 REPeat alea_demo
40  PRINT RND(lin_inf TO lim_sup);" "
;
50 END REPeat alea_demo;" ";
```

La salida de este programa puede ser ésta:

```
105 105 101 105 105 96 100 101 103 95
 103 105 102 102 105 102 96 103 102 9
6 104 96 105 95 100 98 105 97 99 102
```

He aquí un juego breve y fácil de jugar, basado en la producción de números aleatorios. Esta simple ejecución le dará una idea clara de lo que debe hacer:

```
******************************

    Mi primer número es el 6
         El segundo es el 12

         Usted tiene $20

Cuanto apuesta? el próximo
 está entre el 6 y el 12
           10

     Mi número es el 5
   Lo siento, pierde $10


******************************
```

```
      Mi primer número es el 13
           El segundo es el 8

           Usted tiene $10

   Cuanto apuesta? el próximo
    está entre el 13 y el 8?
              3

        Mi número es el 3

      Lo siento, pierde $3

*******************************

      Mi primer número es el 11
           El segundo es el 4

           Usted tiene $7

   Cuanto apuesta? el próximo
    está entre el 11 y el 4
              5

        Mi número es el 8

      Muy bien, gana $10

*******************************

      Mi primer número es el 12
           El segundo es el 7

           Usted tiene $17

   Cuanto apuesta? el próximo
    está entre el 12 y el 7
              4
```

```
        Mi nùmero es el 2

      Lo siento, pierde $4

*******************************

      Mi primer nùmero es el 13
           El segundo es el 8

           Usted tiene $13

Cuanto apuesta? el pròximo
 està entre el 13 y el 8
          2

        Mi nùmero es el 5

      Lo siento, pierde $2

*******************************

      Mi primer nùmero es el 2
           El segundo es el 5

           Usted tiene $11

Cuanto apuesta? el pròximo
 està entre el 2 y el 5
          0

             Cobarde!!

        Mi nùmero es el 2

*******************************

      Mi primer nùmero es el 1
           El segundo es el 6
```

Usted tiene $11

Cuanto apuesta? el próximo
está entre el 1 y el 6
11

Mi número es el 7

Lo siento, pierde $11

Aquí termina el juego

Se ha arruinado!

Y aquí está el listado del programa para que usted y su QL puedan disfrutar de una o dos partidas de *Adivinar el Número*:

```
10 REMark Adivinar el Número
20 REMark Números aleatorios demo
30 CSIZE 1,0
40 PAPER 1:INK 6
50 BORDER 1,2,7
60 CLS:CLS #0
70 d=20
80 REMark ***************
70 REPeat juego
100 CLS
110 a=RND(1 TO 13)
120 b=RND(1 TO 13)
130 IF ABS(b-a)<2 THEN GO TO 120
140 IF ABS(b-a)>7 THEN GO TO 120
150 c=RND(1 TO 13)
160 IF a=c OR b=c THEN GO TO 150
170 PRINT \\\"   Mi primer número es
el ";a
180 PRINT "          El segundo es el "
;b
190 PRINT \\"           Usted tiene $";d
200 PRINT \\"Cuanto apuesta? el próxi
mo"
210 PRINT " está entre el ";a;
```

```
220   PRINT " y el ";b
230 INPUT "              ";e$
240 IF e$="" THEN GO TO 230
250 IF e$<0 OR e$>d THEN GO TO 230
260 e=e$
270   IF e>d THEN GO TO 230
280 FLASH 1:PAPER 5:INK 2
 90 IF e<1 THEN PRINY \\"              Cob
arde!!"
300 PRINT \"    Mi nûmero es ";c;"
    "
310 FLASH 0:INK 6:PAPER 1
320 IF e<1 THEN GO TO 390
330 IF NOT(c>a AND c<b OR c<a AND c>b
) THEN GO TO 390
340 PRINT \"    Muy bien, gana $";2*e
350 d=d+2*e
360 GO TO 390
370 PRINT \"   Lo siento, pierde $";e
380 d=d-e
390 FOR j=1 TO 1000:END FOR j
400 IF d>0 THEN END REPeat juego
410 REMark *****************
420 PRINT \\"        Aquî termina el ju
ego"
430 PRINT \\"          Se ha arruinado!
"\\\\
```

PAUSE

El comando PAUSE se usa cuando se quiere que el programa pare su ejecuciôn durante un tiempo estipulado. La palabra PAUSE va seguida de un nûmero.

```
PAUSE 200
```

Cada unidad de pausa es de 20ms.

Poniendo REMarks

La sentencia REMark se usa para poner una lînea de observaciones o notas dentro de un programa, pero sin que

el ordenador tenga en cuanta esa lînea. He aquî un ejemplo:

```
10 REMark Este programa prueba el RND
```

Una sentencia REMark puede ir detràs de otra sentencia cualquiera dentro de una lînea del programa:

```
100 PRINT "Ha ganado!":REMark Fin juego
```

Sin embargo no puede comenzar una lînea multi-sentencia con una sentencia REMark:

```
450 REMark Final:PRINT "Ha ganado!"
```

La segunda mitad de esta lînea nunca se ejecutarà.

STOP

Como usted puede suponer, STOP se usa para detener un programa:

```
100 ::::
120 IF resp$="Constantinopla" THEN PR
INT "Bien hecho!":STOP
130 PRINT "Esta no es la respuesta qu
e buscaba"
140 :::::
```

INPUT

Esta sentencia se usa para introducir datos (numèricos o cadenas) dentro del programa durante su ejecuciòn en el QL. Su forma es:

```
10 INPUT "Cuàl es tu nombre? ";nom$

10 INPUT "Què edad tienes?;edad
```

Si quiere introducir màs de una entrada en una sentencia INPUT, puede usar los separadores para dar formato a las

correspondientes preguntas:

```
10 INPUT "Cuàntos?"!num,"Edad"!av

10 INPUT "Nombre?"!n$,"Apellido?"!a$
```

INKEY$

INKEY$ se usa para leer las teclas que se pulsen de una en una (o desde un canal que se haya designado). El programa saltarà sobre la lìnea INKEY$ si no se introduce el caràcter que se està buscando:

```
100 REPeat probar
110  IF INKEY$="" THEN EXIT probar
120 END REPeat probar
130 :
140 REPeat probar2
150  b$=INKEY$
160  IF b$<>"" THEN EXIT probar2
170 END REPeat probar2
180 :
190 IF b$="a" THEN PRINT "OK"
```

Este pequeño programa usa el primer INKEY$ para leer el teclado y detiene el programa si no se pulsa una tecla, es decir, si INKEY$ no devuelve una cadena vacìa. Una vez que el teclado està libre, el QL pasa a la lìnea 40, que retiene la acciòn hasta que se pulsa una tecla. Entonces el programa pasa a la lìnea 190. Si la tecla pulsada es la "a" minùscula (Esto es, INKEY$="a") entonces el QL imprime "OK".

Obteniendo LISTados

El comando LIST le permite obtener listados sobre la pantalla, o sobre otro dispositivo, el listado del programa que contiene la memoria del QL.

Se puede usar de cuatro formas diferentes:

LIST 100 TO 120 listarà desde la lìnea 100 hasta la 120

inclusive.

LIST 100 TO lista todas las líneas del programa empezando por la 100 hasta la última línea del programa.

LIST TO 100 empezará desde la primera línea del programa hasta la línea 100 inclusive.

LIST 100 lista solamente la línea 100.

Si no se le especifica un número o rango de líneas, imprime todo el programa desde la primera a la última línea. Si se desea detener el listado en algún punto, basta con pulsar las teclas CTRL y SPACE al mismo tiempo.

Otros Comandos

PEEK y POKE le permiten obtener el contenido de una posición de memoria (PEEK) o modificarlo (POKE). Se usa de la siguiente forma:

```
PEEK (dirección)    obtiene un octeto
PEEK_W (dirección)  obtiene dos octetos
PEEK_L (dirección)  obtiene cuatro octetos

POKE dirección,datos    introduce un octeto
POKE_W dirección,datos  introduce dos octetos
POKE_L dirección,datos  introduce cuatro octetos
```

AUTO se usa para hacer que el QL genere automáticamente los números de las líneas del programa. Para pararlo se pulsan las teclas CTRL y SPACE simultáneamente.

CLEAR se usa para limpiar todas las variables y hacerlas desaparecer. Cuando se ejecuta el comando RUN se obtiene también un CLEAR.

Así como CLEAR se usa para limpiar las variables, NEW se usa para limpiar la memoria de todo su contenido: variables y programa. Debe usarse con cuidado.

Capítulo Diez y Ocho

Manejo de Ficheros

En los cartuchos de 'Microdrive' se pueden almacenar datos permanentemente y leerlos de nuevo cuando se necesiten, dentro de la memoria RAM del QL. El comando que se usa para abrir un fichero es el siguiente:

```
OPEN_NEW #6,MDV1_nombre
```

En este caso se le ha asignado el canal 6 al fichero MDV1_datos. Se puede asignar a un dispositivo cualquier nùmero desde el 3 al 15, siempre que no se lo haya dado anteriormente a otro dispositivo. Los canales 0, 1 y 2 estàn reservados para el QL.

Para introducir datos en un fichero se utiliza el comando PRINT, como si fuera la pantalla, o el INPUT para leer los datos que contiene. Estas dos formas de INPUT son vàlidas:

```
PRINT #6,llenar

INPUT #6,llenar
```

Los datos que se introducen pueden venir de varias fuentes, como de la salida que genera un programa durante su ejecuciòn, de los datos introducidos por el usuario o desde una sentencia DATA que es leida (READ) desde las sentencias del programa.

Una vez que ha introducido sus datos, debe cerrar el fichero con un comando CLOSE, que se usa de la siguiente forma:

```
CLOSE #6
```

Cuando se crea un fichero se abre con OPEN_NEW y una vez que està lleno, se abre con OPEN_IN para poder leer los

datos que contiene, como podrà ver en nuestro siguiente programa. Este programa graba el contenido de la ROM del QL desde la direcciòn 14500 a la 14750 (38A4 a 399E) y ha sido escrito por W. T. Cowhig para la revista Quanta:

```
100 REMark Abriendo y usando ficheros
110 REMark       W T Cowihig, SALE
120 :z
130 CLS:CLS #0
140 k=14500
150 REMark Esto imprime en ASCII
160 :
170 OPEN-NEW #4,mdv1_ASCEE:REMark
    Quite el NEW si el fichero
  ya existe y lo està reusando
180 :
190 FOR N=0 TO 250
200  IF (PEEK(k+N))<=128 THEN b=(PEEK
(k+N)):PRINT !CHR$(b);:PRINT #4,(CHR$
(b);
210  IF PEEK (k+n)>128 THEN c=(PEEK(k
+N)):PRINT !c;:PRINT #4,!c;
220 END FOR N
230 :
240 t$="Esta es la ROM desde 14500 a
14750 -   38A4 a 399E"
250 PRINT t$
260 PRINT #4,t$:CLOSE #4
270 :
```

Capítulo Diez y Nueve

Modelo Financiero

Aunque los cuatro programas que vienen con el QL le servirán para infinidad de cosas, habrá momentos en los que necesitará un programa más simple y directo. Yo necesité un programa de 'prospección de tendencias' y me resultó más fácil escribir el mío propio, que adaptar el material de Psion. Me pareció que este programa podría serle útil a alguien ya que se puede adaptar a una gran cantidad de situaciones diferentes.

Este programa proporciona una versión reducida de algunas de las facilidades de los programas de hoja de cálculo (spread-set). Como podrá ver, el programa toma datos de ventas mensuales o número de devoluciones o cualquier acontecimiento que suceda regularmente y que se haya grabado en intervalos regulares. Con esta información, el programa puede extrapolar las tendencias futuras asumiendo que los factores implicados en la observacion permanecen invariables.

Cuando ejecute el programa, le preguntará si quiere una copia en la impresora. Seguidamente le invitará a introducir el número de meses de los que dispone de datos. Puede cambiar el periodo de tiempo con el que esté trabajando a dias, años, etc. Después le dirá que introduzca los datos relativos a cada mes. El QL realizará los cambios aproximados de mes a mes, comparando el mes dos con el uno, el mes tres con el dos y así sucesivamente, y le dará el cambio de porcentaje medio.

Puede decirle al QL que haga una extrapolación de los resultados, especificando el número de meses para los que quiere los datos de la prospección y si quiere esta prospección basada en el último mes para el que tiene datos o en el resultado medio por mes. Despuès de èsto, puede escojer entre ejecutar la prospección de nuevo desde el

principio o terminar la ejecución.

Aquí está el listado del programa:

```
10 REMark Modelo Financiero
20 CLS:CSLS #0
30  copia
40 REPeat respuesta
50 INPUT \\\"De cuantos meses tiene l
as            figuras disponibles? ";m
60  IF m>1 THEN EXIT respuesta
70 END REPeat respuesta
80 tot=0
90 CLS
100 DIM a(m):DIM b(m)
110 REMark ---------------
120 FOR ak=1 TO m
130 AT 17n0
140 pRINT "
                   "
150 AT 17n0
160 INPUT "Introduzca figura del mes
";(ak);"   ";a(ak)
170 IF ak<17
190  q=ak
190  ELSE q=ak-16
200   IF q>16 THEN q=q-16:GO TO 200
210 END IF
220 AT q,2
230 PRINT "Mes ";ak;" -  ";a(ak)
240 IF z=1 THEN PRINT #3,"  Mes ";ak;
" -  ";a(ak)
250 tot=tot+a(ak)
260 END FOR ak
270 AT 17,0
280 PRINT "
                   "
290 REMark -------------------
300 END IF
310 AT q,0
320 av=tot/m
330 FOR bk=2 TO m
```

```
340   b(bk)=(100-(a(bk-1)*100/a(bk)))
350 END FOR bk
360     PAUSE 100
370 CLS
380 PRINT\"------------------------------
----------"
390 IF z=1 THEN PRINT #3,\"----------
---------------------------"
400 PRINT \"Diferencia entre meses:"
410 IF z=1 THEN PRINT #3,\"Diferencia
 entre meses:"
420     PRINT
430     IF z=1 THEN PRINT #3
440 FOR ak=2 TO m
450 PRINT "  Mes ";ak-1;" a mes ";ak;
 - ";INT(b(ak));"%"
460 IF z=1 THEN PRINT #3," Mes ";ak-1
;" a mes ";ak;" - ";INT (b(ak));"%"
470 END FOR ak
480 REMark ------------------
490     PAUSE 100
500 total=0
510 FOR ak=2 TO m
520  total=total+b(ak)
530 END FOR ak
540 REMark ------------------
550 promedio=INT(total*100/(m-1))/100
560 PRINT \"---------------------------
-----------"
570 IF z=1 THEN PRINT#3,\"-----------
--------------------------"
580 PRINT \" El promedio de cambio es
 ";promedio;"%"
590 IF z=1 THEN PRINT #3,\"  El prome
dio de cambio es ";promedio;"%"
600     PAUSE 100
610 PRINT \"---------------------------
-----------"
620 IF z=1 THEN PRINT #3,\"----------
---------------------------"
630 PRINT \"Ahora la proyección de ca
mbio."
```

```
640 IF z=1 THEN PRINT #3,\"Ahora la p
royección de cambio."
650 INPUT \"Cuantos meses de proyecci
òn                  quiere? ";numero
660 PRINT \"--------------------------
-----------"
670 PRINT El ùltimo mes registrado fu
è ";a(m)
680 PRINT \"El promedio por mes fuè "
;10*(INT (av/10))
690 PRINT \"Quire basar la proyecciòn
 en                   1 - Ultimo mes; o
                      2 - Promedio mensua
l?"
700 REPeat respuesta
710 a$=INKEY$
720 IF a$="1" OR a$="2" THEN exit res
puesta
730 END REPeat respuesta
740 d=a$
750 PRINT \"--------------------------
----------"
760 IF z=1 THEN PRINT #3,\"---------
------------------------"
770 IF d=1 THEN
780  e=a(m)
790  ELSE e=10*INT(av/10)
800 END IF
810 PRINT \"  Mes 1 registrado: ";a(m
)
820 IF z=1 THEN PRINT #3,\"  Mes 1, r
egistrado: ";a(m)
830 REMark ---------------
840 FOR ak=2 TO numero
850  e=e+promedio*e/100
860  PRINT "  Mes ";ak;", proyectado
 ";INT(e)
870  IF z=1 THEN PRINT #3,"  Mes ";ak
;", proyectado ";INT(e)
880 END FOR ak
890 REMark ----------------
900     PAUSE 100
```

```
910 PRINT\"----------------------------
----------"
920 IF z=1 THEN PRINT #3,\"----------
---------------------------"
930 PRINT "Elija entr:"
940 PRINT \"   1 - Ejecutar la proyec
ción"
950 PRINT "   2 - Ejecutar la salida
pero                 sin introducir las
figuras              de nuevo"
960 PRINT "   3 - Ejecutar el program
a desde              el principio"
970 PRINT "   4 - Parar"
980 REMark -----------------
990 REPeat respuesta
1000  a$=INKEY$
1010  IF a$>"0" OR a$<"5"  THEN  EXIT
respuesta
1020 END REPeat
1030 REMark -------------------------
1040 p=a$
1050 SELect ON p
1060 ON p=1
1070   GO TO 630
1080 ON p=2
1090   GO TO 400
1100 ON p=3
1110   RUN
1120 ON p=4
1130   STOP
1140 END SELect
1150 STOP
1160 REMark ------------------------
1170 DEFine PROCedure copia
1180  z=0
1190 PRINT \"Quiere una copia en su i
mpresora?                   (S o N)"
1200 REPeat respuesta
1210  a$=INKEY$
1220  IF a$="S" OR a$="s" OR a$="N" O
R a$="n" THEN EXIT respuesta
1230 END REPeat respuesta
```

```
1240   CLS
1250 IF a$="N" OR a$="n" THEN END DEF
ine copia
1260 REMark Ajustar las dos lîneas
       que siguen, para su impresora
1270    OPEN #3,ser1
1280    BAUD 9600
1290    a=1
1200 END DEFine copia
```

Y èste es un ejemplo del programa en acciòn:

```
  Mes 1 -  1234
  Mes 2 -  1345
  Mes 3 -  1456
  Mes 4 -  1567
  Mes 5 -  1678
  Mes 6 -  1789
  Mes 7 -  1899
  Mes 8 -  1912
  Mes 9 -  2123
  Mes 11 -  2234
  Mes 11 -  2345
  Mes 12 -  2567

---------------------------------------

Diferencia entre meses:

  Mes 1 a mes 2 - 8%
  Mes 2 a mes 3 - 7%
  Mes 3 a mes 4 - 7%
  Mes 4 a mes 5 - 6%
  Mes 5 a mes 6 - 6%
  Mes 6 a mes 7 - 5%
  Mes 7 a mes 8 - 0%
  Mes 8 a mes 9 - 9%
  Mes 9 a mes 10 - 4%
  Mes 10 a mes 11 - 4%
  Mes 11 a mes 12 - 8%

---------------------------------------
```

```
El promedio de cambio es 6.41%

--------------------------------------

Ahora la proyecciòn de cambio.

--------------------------------------

  Mes 1, registrado: 2567
  Mes 2, proyectado: 2731
  Mes 3, proyectado: 2906
  Mes 4, proyectado: 3092
  Mes 5, proyectado: 3291
  Mes 6, proyectado: 3502
  Mes 7, proyectado: 3726
  Mes 8, proyectado: 3965
  Mes 9, proyectado: 4219
  Mes 10, proyectado: 4490
  Mes 11, proyectado: 4778
  Mes 12, proyectado: 5084

--------------------------------------
```

Capítulo Veinte

Perfeccionando sus Programas

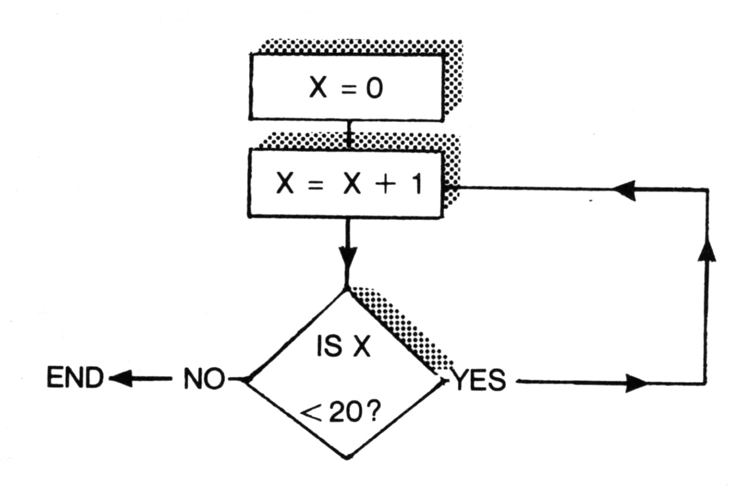
Habrà un punto en su desarrollo como programador del QL, cuando domine el uso de la mayoría de las sentencias del SuperBASIC, que se podrà concentrar en escribir mejores programas, que funcionen correctamente con un mínimo de depuración, que sean fàciles de entender y operar por otras personas y que estèn escritos con lògica y elegancia. Como ya habrà comprobado, el SuperBASIC es una herramienta que le anima al desarrollo de programas estructurados.

Sus programas se ejecutaràn antes y con un mínimo de revisión la primera vez que los introduzca en el QL, si planifica con anterioridad lo que quiere que hagan. Esto es, usted debe pensar en lo que quiere 'sacar' (què es lo que debe aparecer en la pantalla, la impresora y/o los ficheros en 'microdrives'), el modo en que esa salida serà generada y su formato, la 'entrada' de datos y la interacción con el usuario.

Puede comenzar con diagramas de flujo, unas series de figuras unidas por líneas que muestran como serà el flujo de la acción n y las decisiones dentro del QL cuando el programa se estè ejecutando. Las figuras no son demasiado importantes, le sugiero que use solamente dos, un rectàngulo para la mayoría de las acciones que debe ejecutar el QL y un rombo cada vez que el QL tenga que tomar una decisión. Las esquinas del rombo pueden ser usadas, como puede ver en el diagrama, para indicar las alternativas que puede tomar el QL.

La figura nos muestra el diagrama de flujo de un programa que pone la variable X igual a cero y despuès le añade uno. El valor de X es verificado y si es menor que 20, el programa vuelve a añadir uno a X. Esto continúa hasta que el valor de X es igual a 20.

Si tuviera un microordenador con una versiòn de BASIC màs 'normal', podría traducir el diagrama de flujo en el programa que vemos a continuaciòn:




```
10 REMark BASIC ordinario
20 LET x=0
30 LET x=x+1
40 IF x<20 THEN GO TO 30
50 STOP
```

Este programa puede ejecutarse sin dificultades en cualquier ordenador que use el BASIC, incluido el QL. Sin embargo, este programa no està bien estructurado.

Una versiòn del programa para el QL podría ser:

```
10 REMark SuperBASIC
20 x=0
30  REPeat incremento
40   x=x+1
50   IF x=20 THEN EXIT incremento
60 END REPeat incremento
70 STOP
```

Ahora, puede que no capte inmediatamente en què se ha mejorado el programa, incluso tiene dos lîneas màs. Sin embargo, si mira atentamente, verà que tiene unas cuantas ideas ùtiles que pueden resultar provechosas usadas en otros programas que escriba para el QL.

Para empezar, la acciòn se ejecuta mediante un bucle REPeat/END REPeat en vez del GO TO (lînea 40 del original) que se repite hasta que (determinado por la lînea 50) x es igual a 20. Una vez que èsto sucede, se dispara el EXIT y el programa deja el bucle. La ventaja real del programa del QL sobre el otro reside en la posibilidad de hacerlo màs fàcil y claro de entender. El nombre del bucle ('incremento') es perfectamente claro y los lîmites del bucle estàn indicados por el desplazamiento a la derecha de todas las lîneas que lo componen (lîneas 40 y 50).

Un diagrama de flujo nos da un modelo del flujo de la acciòn y de la toma de decisiones dentro del programa. Un programa bien estructurado refleja la claridad de un diagrama de flujo.

El diagrama de flujo es muy ùtil para descubrir errores potenciales en las primeras fases de diseño. Puede encontrar, por ejemplo, que una condiciòn que verifica el programa para tomar la EXIT, no se presentarà nunca, dejando al programa atrapado en un bucle infinito. Otras partes de su programa seràn completamente eludidas debido a que una de las condiciones que dan paso a ese àrea del programa, nunca serà satisfecha.

Una vez que ha proyectado un diagrama de flujo para su programa y lo ha ejecutado mentalmente varias veces de forma que los problemas màs obvios se hayan corregido, debe reducir el diagrama a series de llamadas a procedimientos, si es posible dentro de bucles de REPeat/END REPeat. Esto puede parecer un poco tonto de hacer para un simple programa como el anterior, pero este mètodo demuestra su utilidad en los programas complejos.

Como puede ver, en la mayor parte de los programas de este libro, yo prefiero comenzarlos con series de llamadas a

procedimientos, con cada acciòn del programa ejecutada en un procedimiento diferente. Estos procedimientos llaman a otros procedimientos y pueden incluir repeticiones y asignaciòn de construcciones dentro de ellos. Si los pasos dentro de un programa se van a ejecutar varias veces en una secuencia particular, las series de llamadas a los procedimientos pueden ejecutarse repetidamente hasta que se encuentre una condiciòn concreta que dispare la EXIT para terminar la ejecuciòn.

Usted comprenderà lo ùtil que puede ser esta aproximaciòn a la programaciòn cuando tenga que depurar el programa. Si tiene un error que parece que està dentro de un procedimiento, es realmente fàcil eliminarlo en vez de tener que ejecutar todo el programa hasta que descubra dònde se encuentra el error.

El trabajar con estos 'mòdulos' de procedimientos le permite tambièn verificar secciones de su programa aisladamente antes de que el programa estè totalmente completado. Se lo voy a intentar aclarar con el siguiente ejemplo.

Aquì tenemos las primeras lìneas de un programa imaginario llamado *QL Ajedrez*:

```
100 REMark QL Ajedrez
110 z
120 iniciar
130 :
140 REPeat juego
150  dibujo_tablero
160  acepto_mov_jugador
170  dibujo_tablero
180  QL_mueve
190 IF QL_gana OR humano_gana THEN EX
IT juego
200 END REPeat juego
210 :
220 IF QL_gana THEN
230   PRINT "Yo he ganado, humano!"
240  ELSE
```

```
250   PRINT "Tu eres el ganador!"
260 END IF
270 :
280 STOP
```

Puede tener ejecutàndose el programa en su QL y verificarlo (como un procedimiento de inicio, el que imprime el tablero y el que acepta los movimientos del jugador), antes de dedicar su atenciòn a la parte central donde el ordenador hace su movimiento.

A partir de entonces, no necesita dedicar su tiempo a pensar si el error està en el procedimiento de dibujar el tablero. Habiendo verificado los procedimientos que dibujan el tablero y ejecutan el movimiento del jugador, ya sabe que el error probablemente estarà dentro del procedimiento que realiza el movimiento del ordenador.

Todo lo que tiene que hacer al principio del desarrollo es crear un 'procedimiento falso' que solamente contenga las lìneas DEFine y END DEFine con una lìnea entre ellas que diga algo como PRINT "Aquì mueve el QL". El ordenador aceptarà esa lìnea y cuando ejecute el programa incompleto, le demostrarà si el programa està siguiendo la secuencia correcta, aùn en el caso de que no se hayan escrito los procedimientos vitales.

Observe tambièn, que he usado lìneas en 'blanco' (110, 130, 210 y 270), en las que he puesto solamente dos puntos (:); èsto separa el listado del programa en varias unidades visuales para que sea màs fàcil de estudiar. Hay muy pocas probabilidades de que se le quede corta la memoria del QL, de modo que puede permitirse el lujo de hacer cosas como èsta que, en otros ordenadores, podrìa ser considerado como un lujo. Las lìneas en blanco son, en efecto, màs ùtiles visualmente que las sentencias REMark, pero tienen menos utilidad para resaltar lo que hace esa secciòn del programa. Sin embargo, si ha elegido cuidadosamente los nombres de los procedimientos REPeat, le pueden servir para indicar claramente el propòsito de cada secciòn del programa.

También podrá observar que he separado hacia la derecha las líneas que corresponden a cada bucle REPeat así como en los IF/THEN. Todo esto ayuda a mostrar más claramente lo que está haciendo.

La línea 190 se lee casi como una sentencia en inglés. En ella se hace uso de 'variables Booleanas'. Estas pueden tomar un valor de cero o uno, el cero indica una condición 'falsa' y el uno 'verdadera'. La variable QL_gana, se pone a cero en el procedimiento de iniciación. Lo mismo ocurre con la variable humano_gana. Después, si durante la verifica- ción del ganador (que probablemente formará parte del procedimiento dibujo_tablero), se encuentra que una de estas condiciones ha cambiado, nos indicará que uno de los dos ha ganado. Esto dispara la salida del bucle del juego, yendo a parar al bucle IF/THEN que imprime los mensajes del ganador.

Aunque esta descripción se ha centrado alrededor de un juego, las ideas que se han presentado sirven para cualquier programa que usted escriba en el QL.

Trabaje Primero Sobre Papel

Como le apuntaba al principio de este capítulo, le sugiero que intente hacer tantas pruebas sobre el listado del programa como le sea posible antes de introducirlo en el QL, aunque se sienta tentado a escribirlo directamente sobre el ordenador. Descubrirá que la disciplina de escribirlo antes a mano, le será muy útil y producirá mejores programas que si lo hubiera hecho de la otra forma. Sobre todo, terminará perdiendo menos tiempo que si empieza sentandose al teclado desde un principio.

A mi me costó un tiempo aprender esta lección, a pesar de haber leído en muchos libros "ejecute el programa sobre el papel, del mismo modo que lo haría la máquina, antes de introducirlo en el ordenador", pero normalmente hacía caso omiso de estos consejos.

Debe trabajar, también sobre diagramas de flujo y tener

una idea clara de la organización que se va a dar a la pantalla. Yo no trabajé demasiado sobre el papel hasta que estuve durante dos semanas sin ordenador y con muchas ideas para escribir programas. Entonces tuve que empezar a escribirlos en un cuaderno.

La relativa facilidad con que los programas fueron depurado una vez que los introduje en el ordenador, a pesar de su complejidad (incluido mi primer programa de ajedrez), me convencieron de que éste era el mejor camino para trabajar. Es fantástico lo claro que puede llegar a ser un programa si todo el trabajo de diseño se realiza sobre papel en vez de en la pantalla del ordenador.

Preguntas Explícitas

Cuando se escriben programas es muy útil pensar cómo puede parecer el programa cuando lo vea, por primera vez, una persona ajena. Si se requiere una pregunta de introducción de datos, es mucho más útil que el programa imprima algo parecido a "CUANTAS HORAS HA TRABAJADO EL EMPLEADO ESTA SEMANA?" en vez de "INSERTAR HORAS" o una simple interrogación.

La misma sugerencia es aplicable a los impresos. Es bastante mejor que su programa escriba "EL NUMERO DE HORAS TRABAJADAS A SUELDO COMPLETO ESTA SEMANA SON 27", y no "HORAS, COMPLETAS: 27" o solamente un "27". Por supuesto que, proporcionando preguntas e impresos claros consume memoria, así como tiempo al introducir el programa, pero la contribución que aporta al programa final, hace que el trabajo realizado se de por bien empleado.

Sentencias REM

De la misma forma que las sentencias PRINT y los mensajes de entrada de datos ayudan a la persona que está ejecutando el programa a darse cuenta de lo que está haciendo, las sentencias REM ayudan a presentar el programa más claramente a aquel que está examinando el listado por primera vez.

Las sentencias REM (que, como ya sabe, son ignoradas por el ordenador durante la ejecución) se deben usar para ayudar a aclarar el flujo del programa y lo que sucede en determinados puntos. Son especialmente importantes en las partes del programa donde se toman decisiones o se realizan cálculos.

Variables

Cuando se usan variables es muy útil ponerlas nombres explícitos, usando incluso palabras completas (tal como HORAS para el nombre de una variable en un programa de nóminas para las horas trabajadas) o una versión abreviada de ellas (como HRS) que tienen un significado obvio. Descubrirá lo fácil que resulta así recordar el nombre de las variables. Esto le ayudará en la depuración inicial, así como cuando, posteriormente, quiera ampliar el programa.

Los nombres de variable explícitos le ayudarán a hacer sus programas más 'transparentes' de forma que otros programadores puedan comprender qué es lo que hace cada parte del programa. También le será de gran ayuda a usted mismo, cuando lo vaya a estudiar al cabo de un tiempo. Es sorprendente lo claro que se ve un programa cuando se introduce y lo denso y dificil que resulta cuando se vuelve a estudiar al cabo de algún tiempo.

He aquí un ejemplo de un programa que usa nombres de variables explícitos. Verá lo claro que se ve lo que está sucediendo en cada línea dentro de los bucles FOR/END FOR (líneas 110 a 160), debido a que las variables explican por sí mismas el propósito de cada manipulación aritmética:

```
10 REMark        Interés
20 REMark Simple y compuesto
30 CLS:CLS #0
40 INPUT \\\"  Capital?  ";capital
50 INPUT \\"    Interès?  ";interes
60 INPUT \\"  Por cuantos años?  ";añ
os
70 CLS
```

```
80 PRINT "-----------------------------
---------"
90 PRINT "Años","Simple","Compue.","
 Difer."
100 PRINT "----------------------------
----------"
110 FOR periodo=1 TO años
120 simple=capital+periodo*capital*(i
nteres/100)
130 compuesto=INT(100*capital*(1+inte
res/100)^periodo)/100
140 diferencia=INT(100*(compuesto-sim
ple))/100
150 PRINT periodo,simple,compuesto,;"
   ";diferencia
160 END FOR periodo
```

He aquí dos ejemplos de programas en acción. Observe que los mensajes de entrada de datos, aunque no están excesivamente claros, por lo menos indican al usuario què es lo que se està pidiendo:

```
Capital?  100
   Interès?  8.25
Por cuantos años?  12
```

Años	Simple	Compue.	Difer.
1	108.25	108.25	0
2	116.5	117.18	.68
3	124.75	126.84	2.08
4	133	137.31	4.3
5	141.25	148.64	7.38
6	149.5	160.9	11.39
7	157.75	174.17	16.41
8	166	188.54	22.53
9	174.25	204.1	29.85
10	182.5	220.94	38.44
11	190.75	239.17	48.41
12	199	258.9	59.9

```
Capital?  45
  Interês?  16
Por cuantos años?  5
```

Años	Simple	Compue.	Difer.
1	52.2	52.2	0
2	59.4	60.55	1.15
3	66.6	70.24	3.64
4	73.8	81.47	7.67
5	81	94.51	13.5

Verificaciòn de Datos

Cualquier dato que sea introducido en el programa debe ser verificado antes de aceptarlo, para asegurarse que un dato incorrecto no cause un fallo del programa en algùn punto posterior. Cuando necesite una entrada numèrica, es mejor pedirla como una cadena y, una vez verificada la validez, convertirla en numèrica.

Tambièn debe verificar que todos los datos de entrada produciràn respuestas vàlidas cuando se procesen posteriormente. Por ejemplo, debe asegurarse que su programa no acepta el cero como una posible entrada, cuando posteriormente se va a usar como divisor.

Del mismo modo, si los nùmeros van a ser procesados por una funciòn, y despuès se va a usar el resultado de este proceso para hacer una divisiòn, debe verificar que los datos aparentemente vàlidos, no producen un cero como resultado de la evaluaciòn de esa funciòn.

Si la informaciòn introducida por el usuario es rechazada

y se le pide una nueva entrada, el programa debe darle una idea de por què no ha sido aceptada o preguntarle de nuevo què es lo que quiere exactamente(como "INTRODUZCA UN NUMERO ENTRE 1 Y 4"). Se corre el riesgo de desanimar al usuario si los datos que introduce son rechazados continuamente sin razòn aparente.

Por ejemplo, en nuestro pròximo programa, el ordenador mira si hemos introducido una "f" o una "F" para terminar una cadena de nùmeros (vea las lîneas 110, 160 y 330) una vez que se ha seleccionado una opciòn. La 'secciòn de menù' del programa (*PROCedure acciòn*) necesita un "1" un "2" o un "3" y rechaza otra entrada que no sea esa (vea las lîneas 460 y 470). He aquî el listado del programa:

```
10 REMark  Media aritmètica
20 REMark  y armònica
30 CLS:CLS #0
40 accion
50 REMark ***************
60 DEFine PROCedure aritmetica
70 AT 3,12
80 PRINT "Media Aritmètica"
90 PRINT \"Introduzca los nùmeros a u
sar"
100 PRINT " para hayar la media aritm
ètica"
110 PRINT " >Introduzca 'F' para term
inar          "
120 CSIZE 3,1
130 REPeat entrada
140  INPUT "    ";q$
150     IF q$="" THEN GO TO 140
160     IF q$="f" OR q$="F" THEN EXIT
entrada
170     sum=sum+q$
180     cuenta=cuenta+1
190 END REPeat entrada
200 PRINT \"La media aritmètica es ",
sum/cuenta
210 END DEFine aritmetica
220 REMark ****************
```

```
230 DEFine PROCedure armonico
240 AT 3,12
250 PRINT "Media Armònica"
260 PRINT \\"Introduzca los nùmeros a
 usar"
270 PRINT " para hayar la media armòn
ica"
280 PRINT " >Introduzca 'F' para term
inar         "
290 CSIZE 3,1
300 REPeat entrada
310  INPUT "    ";q$
320     IF q$="" THEN GO TO 310
330     IF q$="f" OR q$="F" THEN EXIT
entrada
340     sum=sum+(1/q$)
350     cuenta=cuenta+1
260 END REPeat entrada
370 PRINT \\"La media armònica es ",1
/(sum/cuenta)
380 END DEFine armonico
390 REMark ***********************
400 DEFine PROCedure accion
410 CSIZE 0,0
420 PRINT \\"Seleccione el programa:"
430 PRINT \"  1 - Media aritmètica"
440 PRINT \"  2 - Media armònica"
450 PRINT \"  3 - fin del programa"
460 z$=INKEY$
470 IF z$<"1" or z$>"3" THEN GO TO 46
0
480 IF z$=3 THEN
490     CSIZE 3,1
500 PRINT \" OK, gracias por su tiemp
o"\\
510     CSIZE 0,0
520    FOR jj=1 TO 1000:END FOR jj
530 END IF
540 CLS
550 cuenta=0
560 sum=0
570 IF z$=1 THEN aritmetica:GO TO 410
```

```
580 IF a$=2 THEN armonico:GO TO 410
590 END DEFine accion
```

Y aquí está el programa en acción:

```
Seleccione el programa:

  1 - Media aritmètica

  2 - Media armònica

  3 - Fin del programa

Media Aritmètica

Introduzca los nùmeros a usar
 para hayar la media aritmètica
 >Introduzca 'F' para terminar

   564.8
   564.5
   56.78
   123.993
   E

La media aritmètica es 327.5182


Seleccione el programa:

  1 - Media aritmètica

  2 - Media armònica

  3 - Fin del programa


Media Aritmètica

Introduzca los nùmeros a usar
 para hayar la media armònica
 >Introduzca 'F' para terminar
```

```
564.8
564.5
56.78
123.993
f

La media armònica es 136.898

Seleccione el programa:

   1 - Media aritmètica

   2 - Media armònica

   3 - Fin del programa

OK, gracias por su tiempo
```

Documentaciòn

La documentaciòn es el material escrito que suele acompañar a un programa. Es de gran utilidad que un programa tenga su documentaciòn, aunque solo sea un bosquejo.

La informaciòn escrita debe explicar, por supuesto, lo que hace el programa y el flujo de la acciòn dentro èste. El programa debe avisar al usuario del tipo de acciones que seràn requeridas por su parte, cuando se ejecute, y darle una indicaciòn de la clase de datos que va a aceptar. Tambièn se debe explicar el formato que tomaràn los datos a la salida.

Si existe algùn tipo de mejora que se puede hacer en el programa, se deben dar sugerencias para ello en la documentaciòn. Tambièn se deben incluir referancias a cualquier material que pueda ayudar a entender los algoritmos usados, o para determinadas àreas del desarrollo del programa.

Es razonable asumir que el trabajo de programaciòn no ha terminado una vez que se ha escrito el programa. Sin la documentaciòn, el trabajo se ha completado solamente en sus tres cuartas partes. La documentaciòn da por terminada la tarea, añadiendo una presentaciòn profesional a su trabajo que permite, al programa que acaba de escribir, ser usado de forma eficiente.

Es mejor intentar escribir la documentaciòn para que dè una idea de su funcionamiento a alguien que no haya visto el programa funcionando. Con la sola lectura de la documentaciòn, esta persona debe ser capaz de obtener una buena idea de como està organizado el conjunto del programa, lo que hace y su interacciòn con el usuario, en terminos de aceptaciòn de datos de entrada y de presentaciòn de los resultados.

La documentaciòn de la mayorìa de los programas suele empezar con una introducciòn, donde se explica ràpidamente què es lo que hace y còmo se usa el programa. En partes posteriores se puede explicar el programa con màs detalle. No es bueno hacer que el usuario navegue a travès de una gran cantidad de informaciòn, para poder enterarse de los aspectos vitales que necesita conocer para hacer funcionar el programa.

Capítulo Ventiuno

La Máquina FORTH

Si quiere puede conseguir una implantación completa del lenguaje FORTH para el QL (de Computer One, Science Park, Milton Road, Cambridge. 0243-862616). Sin embargo, si solamente quiere una pequeña idea de algunos aspectos del FORTH, puede usar el programa que le damos en este capítulo que le permitirá experimentar con un 'mini FORTH'.

El FORTH, fuè desarrollado por Charles Henry Moore, que se sentía frustrado con los lenguajes de programación que usaba normalmente (tales como el FORTRAN y el ALGOL) para escribir programas para controlar radio telescopios. El Sr Moore desarrolló primero una versión primitiva de este lenguaje en los primeros años de la dècada de los setenta, y finalmente consiguió una versión integrada del lenguaje funcionando en un IBM 1130, el ordenador màs potente de que disponía. El lenguaje se llamó FORTH (contracción de fourth, cuarto en inglès) porque Moore estaba trabajando con ordenadores de la 'tercera generación' y veía este lenguaje como una 'cuarta generación'. Sin embargo, el 1130 solo permitía nombres de ficheros con cinco caracteres, por lo que la palabra 'fourth' se abrevió como FORTH. Como apunta Moore en la introducción del libro Starting FORTH (FORTH Inc., Leo Brodie, Prentice-Hall, 1981), también se hizo porque comparado con 'fourth', FORTH es 'un divertido juego de palabras'.

Lo mejor del FORTH, es que viene equipado con un grupo de comandos estàndar que se usan para desarrollar sus propios comandos, que a su vez pueden ser usados en la construcción de comandos posteriores. Esto lo hace ràpido, potente y flexible.

Una de las cosas menos usuales del FORTH es su aritmètica, aspecto del FORTH que intenta emular el primer programa de este capítulo. Cuando quiere sumar 50 y 9 en el QL, intro-

duce lo siguiente:

```
PRINT 50 + 9
```

Para hacerlo en FORTH necesita introducir:

```
59 9 + .
```

Primero introduce el primer nùmero, despuès un espacio, luego un segundo nùmero y otro espacio y finalmente la operaciòn que desea realizar (en este caso '+' para sumar) seguido de un "." que hace que el programa imprima el resultado. Si no introduce el ".", el ordenador darà también el resultado, pero no lo podrà ver.

Un ordenador FORTH està orientado al "stack" o pila. La pila es como un montòn de papeles. Usted puede escribir en un trozo de papel y ponerlo encima del montòn, o puede cogerlo, pero solo puede retirar un papel de lo alto del montòn. El ùltimo papel que ponga en la pila debe ser el primero que retire.

Cuando introduce el 50 seguido de un espacio, el ordenador lo escribe en un trozo de papel y lo coloca en lo alto de la pila. Despuès escribe el 9 en otro trozo y lo coloca encima del 50, de modo que el 50 es ahora el segundo en la pila. Despuès, el ordenador encuentra la instrucciòn "+" que es una palabra de su "diccionario" y sabe que "+" le ordena que coja dos nùmeros de lo alto de la pila , los sume y coloque el resultado en lo alto de la pila. Esto es lo que hace nuestro programa "50 3 + .". Finalmente el ordenador ve el "." y lo usa como una señal para "coger el valor de lo alto de la pila" o -como en este caso- imprimirlo. Si no incluye un "." el resultado permanecerà en la pila. Puede probar esto introduciendo una operaciòn aritmètica sencilla sin poner el "." para ver que sucede. Pare luego el programa (pulsando la tecla ENTER sin introducir nada màs) y pregunte al QL, en modo directo:

```
PRINT pila
```

Cuando haga esto, èl imprimirà el resultado que tiene en

la pila. La explicaciòn puede perecer un poco complicada, pero le quedarà clara cuando ejecute su programa. Antes de seguir adelante, veamos algunos ejemplos:

? 3 2 + .	? 3 2 + .
Pila=5	Pila=5
? -3 -2 + .	? -3 -2 + .
Pila=-5	Pila=-5
? 3 2 - .	? 3 2 - .
Pila=1	Pila=1
? -3 -2 - .	? -3 -2 - .
Pila=-1	Pila=-1
? 3 -2 -	? 3 -2 -
? 3 2 * .	? 3 2 * .
Pila=6	Pila=6
? 7 4 / .	? 7 4 / .
Pila=1	Pila=1
? 4 7 / .	? 4 7 / .
Pila=0	Pila=1

```
? 77 9 / .                   ? 77 9 / .

  Pila=8                       Pila=8


? 0 9 / .                    ? 0 9 / .

  Pila=0                       Pila=0


? 9 0 / .                    ? 9 0 / .
División por cero es División por cero es
ilegal                       ilegal

? 8 8 * .                    ? 8 8 * .

  Pila=64                      Pila=64


? k                          ? k
```

Y èste es el programa con el que puede experimentar por usted mismo con la Notaciòn Polaca Inversa:

```
100 REMark     'Aritmètica FORTH'
110 REMark (Notaciòn Polaca Inversa)
120 :
130 PAPER 1:INK 7
140 CLS:CLS #0
150 pila=0
160 :
170 REPeat forth
180  PRINT \\
190   INPUT " ? ";f$
200   IF LEN(f$)<2 THEN EXIT forth
210   f$=" " & f$
220   q=LEN(f$)
230   salida=0
240 :
250   IF f$(q)="." THEN
260    salida=1
270    f$=f$(1 TO q-2)
```

```
280    q=q-2
290   END IF
300 :
310   j=q
320   IF f$(j)="*" OR f$(j)="/" OR f$(
j)="+" OR f$(j)="-" THEN
330   flag=(f$(j)="*")+2*(f$(j)="/")+3
*(f$(j)="+")+4*(f$(j)="-")
340    x=j-1:z=j-2
350    REPeat extraccion
360     x=x-1
370     IF f$(x)=" " THEN EXIT extracc
ion
380    END REPeat extraccion
390    segundo$=f$(x+1 TO z)
400   END IF
410 :
420   f$=f$(1 TO x-1)
430   q=LEN(f$)
440   q=LEN(f$)
440   z=q
450 :
460   REPeat extraccion2
470    q=q-1
480    IF f$(q)=" " THEN EXIT extracci
on2
490   END REPeat extraccion2
500 :
510   primero$=f$(q+1 TO z)
520 :
530   IF segundo$="0" AND flag=2 THEN
540    PRINT " División por cero es il
egal"
550    END REPeat forth
560   END IF
570 :
580   IF flag=1 THEN pila=INT(INT(prim
ero$)*INT(segundo$))
590   IF flag=2 THEN pila=INT(INT(prim
ero$)/INT(segundo$))
600   IF flag=3 THEN pila=INT(primero$
+segundo$))
```

```
610  IF flag=4 THEN pila=INT(primero$
-segundo$))
620  IF salida THEN PRINT \"   Pila=
";pila
630 :
640 END REPeat forth
```

El uso del INT en las operaciones aritmèticas, se debe a que el FORTH trabaja solamente con nùmeros enteros.

La Màquina

Ahora que la 'aritmètica FORTH' està comprendida, se darà cuenta que està limitada innecesariamente. Por lo tanto, me impuse la tarea de escribir un programa que permitiera emular muchos otros aspectos del FORTH. Nuestro pròximo programa, la *Màquina FORTH*, es el resultado de esta decisiòn.

Como dijimos antes, el FORTH està "orientado a la pila". Este programa tiene una pila de 20 elementos, aunque en la pantalla solamente aparezcan los 10 elementos superiores. Despuès de cada operaciòn, la pila es reescrita. A su izquierda hay una copia de los 10 elementos de la pila, tal y como estaban antes de la operaciòn, de forma que pueda ver el efecto que tiene cada operaciòn. Con la ayuda de la *Màquina FORTH* y un libro de FORTH (para èsto, puede encontrar interesante mi libro *Learn FORTH on your BASIC micro*), puede aprender una gran cantidad de FORTH y experimentar con muchos de sus elementos.

Bàsicamente, las operaciones del FORTH se realizan poniendo los nùmeros en la pila ('pushing') o tomàndolos de ella ('popping'). La *Màquina FORTH* hace uso casi constante de los dos procedimientos: push_stack y pull_stack. Otro procedimiento, pop_stack, se usa para tomar el valor de lo alto de la pila e imprimirlo en la pantalla.

Lo màs apasionante del FORTH es la facilidad con que se pueden definir nuevas palabras. La palabra DUP duplica el valor que hay en lo alto de la pila. La palabra "." toma el

número de lo alto de la pila y lo imprime. El punto es destructivo, esto es, se pierde el número que usa. Por lo tanto, si usa DUP antes del "." puede conseguir imprimir el número y conservarlo en la pila para su uso posterior. Nosotros vamos a definir una palabra para nuestro propio uso -PRINTOUT- que duplica el número de lo alto de la pila, luego lo toma y lo imprime. Para crear palabras debe proceder como sigue:

```
: PRINTOUT DUP . ;
```

Debe empezar con ":" seguido de un espacio (en FORTH los espacios son vitales) y despuès poner las palabras que quiere incluir en su definiciòn(en este caso DUP .), seguidas por un ";" como señal de que se ha terminado de definir la palabra. De ahora en adelante solamente tiene que poner PRINTOUT en sus programas FORTH y el ordenador realizarà automàticamente "DUP ." por usted. La *Màquina FORTH* puede manejar 20 nuevas palabras definidas por el usuario (sin embargo, puede manejar màs de 20 con unos simples cambios, pero la idea original de este programa es permitirle experimentar con el FORTH màs que crear un FORTH completo).

Si quiere definir una palabra que calcule el cuadrado del número que està en lo alto de la pila, hacer una copia del resultado e imprimir la respuesta, puede definir una palabra como la que sigue:

```
:SQUARE DUP * DUP . ;
```

En esta definiciòn, SQUARE es el nombre de la palabra que està definiendo. DUP duplica el número de lo alto de la pila y usa "*" para multiplicarlo por sí mismo. La respuesta se coloca en lo alto de la pila y "DUP ." copia la respuesta y la imprime, dejando la copia de la respuesta en la pila.

Vamos a ver ahora una parte del FORTH aún màs interesante. Observarà que la última parte de la definiciòn de SQUARE (DUP .) es la misma que la definiciòn de PRINTOUT. Por què no usar PRINTOUT en la definiciòn de SQUARE? No hay ninguna

razòn por la que no lo podamos hacer. La verdadera magia del FORTH es que puede añadir màs y màs palabras a su diccionario, que estèn compuestas por otras palabras que ha definido previamente. Como el QL salva las variables con el programa, puede tener las palabras definidas listas para su uso en cualquier momento. El programa incluye una opciòn para arrancar en 'frio' o en 'caliente'. En caliente preserva el contenido de su diccionario y de la pila.

Pero, por el momento, volvamos a la definiciòn de SQUARE. Recuerde que PRINTOUT es DUP . y que SQUARE es DUP * DUP . por lo que podemos redefinir SQUARE asì:

```
SQUARE DUP * PRINTOUT ;
```

Esto lo puede hacer con la *Màquina FORTH*, incluyendo en la definiciòn una o dos palabras definidas previamente. Si quiere cambiar la definiciòn de una palabra, puede usar el comando 'FORGET' para borrar la palabra del diccionario, antes de volverla a definir. (Aunque borre la palabra que usted le diga, no borrarà otras palabras que incluyan a èsa en su definiciòn).

El FORTH en Acciòn

El FORTH trabaja màs o menos asì:

--El usuario teclea algo en el ordenador
--El FORTH mira en su diccionario para ver si alguna de las palabras coincide con las que ha tecleado.
--Si la encuentra en el diccionario del programa o en el de palabras definidas por el usuario, ejecuta la palabra y vuelve al siguiente elemento de la entrada.
--Si no la encuentra, asume que la palabra es un nùmero.

Nuestra *Màquina FORTH* trabaja de forma similar. Primero busca en la entrada las palabras definidas, cambiàndolas por sus elementos primitivos, despuès trabaja con la entrada procesàndola por orden. Si un elemento de la entrada no es ni una palabra ni un nùmero, lo ignora (la mayorìa de los FORTH dan un mensaje de error). Cualquier nùmero que se encuentre es colocado en la pila, mientras

que las operaciones aritmèticas son ejecutadas.

Verà el progreso del programa, a lo largo de la entrada, por una lînea roja que aparece sobre lo que està procesando. Su trabajo con la entrada lo realiza de izquierda a derecha segùn lo va procesando, de forma que pueda usted ver lo que està sucediendo.

No voy a intentar enseñarle mucho FORTH en este capîtulo. La tarea requiere un libro completo, por lo que he incluido al final del capîtulo una lista de los libros que pueden serle ùtiles para aprender màs sobre el FORTH. En su lugar, le voy a explicar brevemente lo que hacen las palabras que tiene definidas. Puede experimentar con ellas para usar el programa en toda su potencia. Una vez que haya introducido la *Màquina FORTH* en su QL, verà que mientras experimenta con los distintos elementos del lenguaje y crea los suyos propios, va comprendiendo ràpidamente lo apasionante que puede ser el FORTH.

Hay algunas restricciones en el uso de nuestro FORTH. cualquier nombre (de hasta 400 caracteres de longitud) es vàlido para una palabra, con sòlo dos restricciones. Primero el nombre no puede contener un espacio, puede nombrar nuevas palabras con nombres que ya estèn en el diccionario, y puede combinar sîmbolos, nùmeros y caracteres en cualquier combinaciòn (asî 6gG@*h es un nombre vàlido). Si se siente diabòlico puede definir la multiplicaciòn como una divisiòn, o 7 como -184, aunque puede que su ordenador se vea un tanto extraño si lo hace.

Segundo, No debe usar el nombre de una palabra ya definida, como parte del nombre de otra. Esto es, si llama a una palabra TOTAL, no llame a otra SUBTOTAL, aunque sî puede llamrla SUBTOTA, o SUBtOTAL (la Màquina FORTH distingue entre palabras con mayùsculas y minùsculas. Para las palabras FORTH puede usar mayùsculas o minùsculas, o una combinaciòn de ambas.

La gran utilidad de definir palabras se puede demostrar con el siguiente ejemplo. Suponga que quiere una palabra que se llame CIRCLE que debe tomar un nùmero de la pila y

tratarlo como el diàmetro de un cìrculo y dibujar la circunferencia. Lo ùnico que necesitarà introducir es algo como 23 CIRCLE para conseguir dibujar la circunferencia de diàmetro 23 unidades.

La palabra CIRCLE se puede definir como sigue (donde 355/113 se usa como una aproximaciòn entera del nùmero PI):

```
: CIRCLE 355 113 */ . ." es la
circunferencia" ;
```

Entonces, solo necesitarà introducir un nùmero ,o dejar al ordenador que lo tome de la pila, para el diàmetro y su *Màquina FORTH* harà el resto. Verà lo entretenido que es definir palabras en FORTH.

Hay dos versiones estàndar de FORTH, fig-FORTH y FORTH-79. La *Màquina FORTH* se adhiere al FORTH-79 estàndar, pero permite usar equivalentes del fig-FORTH. En algunos casos, la palabra MINUS del fig-FORTH puede intercambiarse con la palabra NEGATE del FORTH-79. Aquì tenemos el vocabulario suministrado con nuestro programa:

+ Toma los dos nùmeros de lo alto de la pila, los suma y coloca el resultado en la pila.

- Toma los dos nùmeros de lo alto de la pila, resta el primero del segundo y coloca el resultado en la pila.

* Toma los dos nùmeros de lo alto de la pila, los multiplica y coloca el resultado en la pila.

/ Toma los dos nùmeros de lo alto de la pila, divide el segundo por el primero y coloca el resultado en la pila.

MOD Realiza una divisiòn como el / pero coloca el resto en la pila.

/MOD Tambièn divide pero poniendo el cociente y el resto en la pila.

*/ Hace una multiplicación y después una división dejando el cociente. Necesita tres números de la pila.

*/MOD Es igual que */ pero devuelve el cociente y el resto, con el cociente en lo alto de la pila.

** Toma dos números de la pila y eleva el segundo número a la potencia del primero, colocando el resultado en la pila.

NEGATE Esta es una palabra del FORTH-79 que multiplica el número de lo alto de la pila por menos uno, cambiando su signo. El equivalente en fig-FORTH es MINUS que también es válido para nuestra *Máquina FORTH* (vea la línea 830).

ABS Realiza la misma función que el ABS del BASIC, devolviendo el valor absoluto del número de lo alto de la pila.

MAX y MIN Este par de palabras compara los dos elementos de lo alto de la pila, dejando solamente el mayor o el menor.

DUP Significa 'duplicar' y hace una copia del número de lo alto de la pila poniéndolo encima del número original.

?DUP Funciona como DUP, pero solo si la pila no está vacía. El equivalente en fig-FORTH es -DUP, que también tenemos definido.

OVER Toma el segundo número de la pila, lo copia y lo pone en lo alto de la pila. De forma que si el número de arriba es A y el segundo B, la palabra OVER haría que la pila se leyera A B A, de arriba hacia abajo.

PICK Debe ser precedido por un número y selecciona el elemento correspondiente a ese número en la pila (si el número que lo precede es el 6, seleccionará

el sexto elemento, contando hacia abajo), copiàndolo en lo alto de la pila.

DROP Borra el nùmero de lo alto de la pila.

SWAP Hace que se intercambien los dos nùmeros de lo alto de la pila.

ROT De rotar, toma el tercer elemento de la pila y lo coloca en primer lugar desplazando hacia abajo el primero y el segundo.

ROLL Va precedido por un nùmero y toma el elemento correspondiente a ese nùmero y lo coloca en la primera posiciòn borràndolo de su posiciòn original y moviendo los otros elementos una posiciòn hacia abajo.

DEPTH Nos dice el nùmero de elementos que hay en la pila. En la *Màquina FORTH* hay un màximo de 20.

VLIST Lista el vocabulario del programa, con las palabras definidas por el usuario en primer lugar, desde la màs nueva a la màs vieja.

FORGET Debe ir seguido por el nombre de una palabra que haya definido el usuario. Borra esa palabra del diccionario, pero no destruye las palabras definidas despuès de esa. El FORTH original destruye todas las palabras definidas detràs de ella.

." Se usa para preceder a un texto de salida. Debe ir seguido por un espacio. La *Màquina FORTH* toma como texto todo lo que sigue hasta las siguientes comillas, que pueden seguir al texto sin ningùn espacio entre medias. Esto significa que ."HOLA" imprimirà la palabra HOLA.

Se le llama 'tick' y se usa para ver si existe una palabra determinada en el diccionario. Debe introducir 'nombre y el programa responderà con nombre

OK si la palabra existe en el diccionario.

KEY Funciona como INKEY$ en BASIC, esperando hasta que se pulse una tecla y poniendo el còdigo ASCII del caràcter en lo alto de la pila. Algunos FORTH lo ponen como INKEY. Esta ùltima no està incluida en el nuestro. Usted la puede añadir si quiere.

EMIT Es el opuesto de KEY e imprime el caràcter correspondiente al còdigo ASCII que le precede, de forma que si escribe 42 EMIT, se imprimirà un asterisco.

SPACES Puede ser usado para dar formato a la salida, ya que toma el nùmero de lo alto de la pila e imprime ese nùmero de espacios.

DO LOOP Es, en cierto modo, como un bucle FOR/NEXT de BASIC. Repite todo lo que viene entre las palabras DO y LOOP. El nùmero de repeticiones es la diferencia entre los dos nùmeros de lo alto de la pila, donde el segundo nùmero debe ser mas alto que el primero. Esto es, si el segundo nùmero de la pila es 20 y el primero 10, el bucle se repetirà diez veces.

Hay dos palabras adicionales en el vocabulario del programa que tienen un significado diferente al del FORTH-79.

LIST Esta palabra se usa en FORTH para controlar los discos. En nuestro FORTH, LIST va seguida por el nombre de una palabra definida por el usuario e imprime el significado asignado a esa palabra, de modo que puede verificar una palabra si ha olvidado el significado que le diò.

SP! Esta es una palabra del fig-FORTH que se usa para iniciar el apuntador de la pila. En nuestro FORTH, SP! vacìa toda la pila.

Las siguientes salidas impresas le pueden dar una idea de lo que hace la *Màquina FORTH* cuando està funcionando.

Primero le da la opciòn de comenzar, destruyendo o conservando el contenido actual de la pila y las palabras definidas (esta informaciòn no se pierde introduciendo RUN en el QL):

```
      Pulse 1 para arranque en caliente
            2 para arranque en frio

        2

      MAQUINA FORTH iniciandose
```

Si pulsa "2", le està diciendo al programa que se inicie. Entonces imprimirà la pila lîmpia y el mensaje "OK", que siempre indica que el FORTH està dispuesto a aceptar entradas:

```
       Antes    Despuès
       , _ ,    , _ ,
       , _ ,    , _ ,
       , _ ,    , _ ,
       , _ ,    , _ ,
       , _ ,    , _ ,
       , _ ,    , _ ,
       , _ ,    , _ ,
       , _ ,    , _ ,
       , _ ,    , _ ,
       , _ ,    , _ ,
       OK
```

Introducimos un grupo de comandos siguiendo el mensaje "?" y el FORTh imprime de nuevo los datos introducidos, despuès de haber cambiado todas las palabras definidas por el usuario por sus elementos primitivos. En este caso, introducimos cuatro nùmeros, solicitando una multiplicaciòn y divisiòn modular y despuès tomamos los dos nùmeros de lo alto de la pila. Puede ver, siguiendo los impresos, lo que sucede en la pila:

```
      ------------------------------------

      ? 8 7 9 4 */mod . .
```

```
8 7 9 4 */mod . .

 Antes    Después
, _ ,        8
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,

 Antes    Després
  8          7
, _ ,        8
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,

 Antes    Después
  7          9
  8          7
, _ ,        8
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,

 Antes    Después
  9          4
```

```
   7           9
   8           7
 , _ ,         8
 , _ ,       , _ ,
 , _ ,       , _ ,
 , _ ,       , _ ,
 , _ ,       , _ ,
 , _ ,       , _ ,
 , _ ,       , _ ,

  Antes    Despuès
   4           9
   9           7
   7           8
   8         , _ ,
 , _ ,       , _ ,
 , _ ,       , _ ,
 , _ ,       , _ ,
 , _ ,       , _ ,
 , _ ,       , _ ,
 , _ ,       , _ ,
4

  Antes    Despuès
   9           7
   7           8
   8         , _ ,
 , _ ,       , _ ,
 , _ ,       , _ ,
 , _ ,       , _ ,
 , _ ,       , _ ,
 , _ ,       , _ ,
 , _ ,       , _ ,
 , _ ,       , _ ,
9

  Antes    Despuès
   7           7
   8           8
 , _ ,       , _ ,
 , _ ,       , _ ,
 , _ ,       , _ ,
```

```
  , _ ,      , _ ,
  , _ ,      , _ ,
  , _ ,      , _ ,
  , _ ,      , _ ,
  , _ ,      , _ ,
OK
```

Como puede ver, el 7 y el 8 son colocados en la pila. Introducimos un nuevo grupo de comandos. Primero le decimos que imprima las palabras "ahora gira" y despuès que haga un ROT, un DUP y despuès imprima. Veamos estos comandos en acciòn:

```
? ."ahora gira" rot dup .

  ."ahora gira" rot dup .

ahora gira

 Antes    Despuès
  7        , _ ,
  8         7
, _ ,       8
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
, _ ,      , _ ,
Pila vacìa
```

El ROT pone un elemnto vacìo en lo alto de la pila. Este es duplicado e impreso, resultando el mensaje "Pila vacìa".

A continuaciòn le decimos al FORTH que imprima "Gira de nuevo" seguido por SWAP y ROT, poniendo 5 y 4 en la pila y multiplicàndolos entre sì:

```
? ."gira de nuevo" swap rot 5 4 *

   ."gira de nuevo" swap rot 5 4 *

 gira de nuevo

  Antes    Despuès
   7         8
   8         7
 ' _ '     ' _ '
 ' _ '     ' _ '
 ' _ '     ' _ '
 ' _ '     ' _ '
 ' _ '     ' _ '
 ' _ '     ' _ '
 ' _ '     ' _ '
 ' _ '     ' _ '

  Antes    Despuès
   8       ' _ '
   7         8
 ' _ '       7
 ' _ '     ' _ '
 ' _ '     ' _ '
 ' _ '     ' _ '
 ' _ '     ' _ '
 ' _ '     ' _ '
 ' _ '     ' _ '
 ' _ '     ' _ '

  Antes    Despuès
 ' _ '       5
   8       ' _ '
   7         8
 ' _ '       7
 ' _ '     ' _ '
 ' _ '     ' _ '
 ' _ '     ' _ '
 ' _ '     ' _ '
 ' _ '     ' _ '
 ' _ '     ' _ '
```

```
 Antes     Despuès
   5          4
, _ ,         5
   8        , _ ,
   7          8
, _ ,         7
, _ ,       , _ ,
, _ ,       , _ ,
, _ ,       , _ ,
, _ ,       , _ ,
, _ ,       , _ ,

 Antes     Despuès
   4          5
   5        , _ ,
, _ ,         8
   8          7
   7        , _ ,
, _ ,       , _ ,
, _ ,       , _ ,
, _ ,       , _ ,
, _ ,       , _ ,
, _ ,       , _ ,

 Antes     Despuès
   5        , _ ,
, _ ,         8
   8          7
   7        , _ ,
, _ ,       , _ ,
, _ ,       , _ ,
, _ ,       , _ ,
, _ ,       , _ ,
, _ ,       , _ ,
, _ ,       , _ ,

 Antes     Despuès
, _ ,         20
   8        , _ ,
   7          8
, _ ,         7
, _ ,       , _ ,
```

```
, _ ,        , _ ,
, _ ,        , _ ,
, _ ,        , _ ,
, _ ,        , _ ,
, _ ,        , _ ,
OK
```

Ahora intentaremos definir una pocas palabras. Primero definiremos PRINTOUT que es, como dijimos anteriormente, un DUP con impresiòn:

```
: printout dup . ;
```

Podemos usar esta palabra dentro de la definiciòn de otras palabras. Observe como nuestro FORTH cambia los elementos de otras palabras definidas por sus formas primitivas antes de compilar la nueva palabra:

```
? : numeros 3 imprimir 2 imprimir ;

? : numeros 3 dup . 2 dup . ;
```

Lo probaremos tecleando NUMEROS y viendo los resultados:

```
------------------------------------------

? numeros

  3 dup . 2 dup .

 Antes    Despuès
  20        3
, _ ,       20
  8       , _ ,
  7         8
, _ ,       7
, _ ,     , _ ,
, _ ,     , _ ,
, _ ,     , _ ,
, _ ,     , _ ,
, _ ,     , _ ,
```

```
 Antes     Despuès
  3          3
  3          20
  20       ' - '
' - '        8
  8          7
  7        ' - '
' _ '      ' _ '
' _ '      ' _ '
' _ '      ' _ '
' _ '      ' _ '
3

 Antes     Despuès
  3          2
  20         3
' - '        20
  8        ' - '
  7          8
' - '        7
' _ '      ' _ '
' _ '      ' _ '
' _ '      ' _ '
' _ '      ' _ '

 Antes     Despuès
  2          2
  2          3
  3          20
  20       ' - '
  - '        8
  8          7
  7        ' - '
' _ '      ' _ '
' _ '      ' _ '
' _ '      ' _ '
2

 Antes     Despuès
  2          2
  2          3
  3          20
```

```
  20          ' _ '
' _ '           8
  7           ' _ '
' _ '         ' _ '
' _ '         ' _ '
' _ '         ' _ '
OK
```

Para verificar que *numeros* e *impreso* han entrado realmente en el vocabulario del programa, introduzca *VLIST*

```
? vlist

  vlist

numeros impreso         +       -
   *        /       MOD     /MOD    */
*/MOD   **      ."      NEGATE  MINUS
   DEPTH   ABS     MIN     MAX     SW
AP
FORGET  VLIST   DUP     ?DUP    -DUP
   ROT     EMIT    SPACES  ROLL    OV
ER
'       DROP    PICK    LIST    KEY
   SP! DO LOOP
```

Un Juego

Las palabras que hemos definido hasta ahora, no son de mucha utilidad por sí mismas así que, he decidido ser un poco mas ambicioso e intentar crear un juego. Con un lenguaje tan limitado, sin facilidades para crear números aleatorios, y sin màs entrada a la *Màquina FORTH* que la palabra KEY, los juegos que se pueden producir son bastante limitados. En este programa la *Màquina* "piensa" un número entre cero y nueve, y usted tiene cuatro oportunidades para

adivinarlo. (Se supone que usted no va a mirar el nùmero en lo alto de la pila) Introducirà sus intentos mediante las teclas numèricas del QL, y èl le darà una respuesta como un nùmero positivo o negativo, o un cero para el caso de que haya acertado el nùmero. Este juego es de poca utilidad, pero nos demuestra la potencia del FORTH.

Primero definiremos la palabra START:

```
start  7 ." key " key 48 - ." key?"
key 48 - swap
```

Esta palabra pone un siete en la pila, imprime la palabra TECLA, acepta un caracter numèrico y le resta 48 a su còdigo ASCII (de forma que "1", cuyo còdigo es 49, pasarà a ser 1 y asî sucesivamente), imprime la palabra KEY? y acepta otro dato numèrico. Los dos nùmeros son intercambiados con SWAP.

La segunda palabra es INICIALIZA:

```
inicializa      dup dup rot swap pick
```

Con èsta, el nùmero de lo alto de la pila es duplicado dos veces, hace una rotaciòn y despuès un PICK. Esto asegura que en lo alto de la pila tenemos ahora un nùmero aleatorio.

Ya tenemos en su lugar las condiciones de empieze. Cada 'movimiento' en el juego es realizado con la palabra JUGAR:

```
jugar   dup ." ahora! " key 48 - - .
```

Esta duplica el nùmero de lo alto de la pila e imprime el mensaje AHORA!, acepta una entrada, le resta 48 y resta èste nùmero al de lo alto de la pila. Este nùmero es un duplicado del que ha pensado la *Màquina* de modo que si el nùmero que resulta es cero, el jugador ha introducido ese mismo nùmero.

Todas estas palabras pueden ser combinadas con un bucle DO LOOP, en una palabra llamada JUEGO:

```
: juego start inicializa 5 1 do jugar
loop
```

Cuando compile JUEGO, se convertirà en esto:

```
juego    7 ." key " key 48 - ." key?"
key 48 - swap dup dup rot swap pick 5
 1 do dup ." ahora! " key 48 - - . lo
op
```

Esto le da cuatro oportunidades para adivinar el nùmero. Estas series de palabras no forman un buen juego, pero sirven para demostrar clàramente còmo se pueden definir las palabras y còmo combinarlas en nuevas definiciones. Tambièn le resultarà ùtil como punto de partida para desarrollar sus propios juegos y rutinas similares.

Definiendo Palabras

Antes de llegar al listado de la *Màquina FORTH*, veamos con màs detalle la forma de definir nuevas palabras. Hay cuatro partes en una palabra. Imagìnese que èsta es una palabra definida:

```
STARS 42 emit 42 emit 1 1 . . ;
```

La definiciòn de la palabra empieza con dos puntos, que le dice a la *Màquina* que pase a modo compilador. Despuès de un espacio va el nombre que le vamos a asignar a la palabra. A continuaciòn va la definiciòn. En este caso la definiciòn es 42 EMIT 42 EMIT 1 1 .. que imprimirà dos asteriscos, colocando el nùmero uno dos veces en la pila e imprimiendo finalmente los dos nùmeros. La parte final de la definiciòn es un punto y coma que le indica a la *Màquina* que se ha terminado la defini- ciòn. No puede compilar palabras nuevas en la misma entrada que otras acciones, sin embargo puede mezclar las palabras nuevas con las otras del vocabulario inicial del programa. Tambièn puede usar sus definiciones dentro de otras definiciones, de modo que lo siguiente es perfectamente vàlido:

```
: GALAXIA STARS STARS ;
```

Puede borrar la palabra STAR con el comando *FORGET STARS* y verificarlo con *VLIST* para comprobar que ha desaparecido del diccionario, aùnque (al contrario que en el verdadero FORTH) GALAXIA todavía funcionarà.

La *Màquina* no proporciona constantes ni variables, pero puede simularlas asignando un valor a una palabra y usandola como un nùmero, borrandola y definiendola de nuevo para cambiar su valor. Por lo tanto:

```
: prueba 65 ;
```

darà el valor 65 a PRUEBA y desde èste momento, se puede usar como si fuera realmente el nùmero 65. Para cambiar este valor, debe hacer FORGET PRUEBA seguido de: *TESTING valor;* para asignarle un nuevo valor. Observe que cuando hace FORGET de una palabra, no recupera el espacio, de modo que no se pueden hacer màs de 20 definiciones en una serie de arranques en caliente. Si intenta crear màs de 20 palabras, la palabra nueva tomarà el lugar de la veinteava palabra que haya definido.

Ahora, por fin, aquì tiene el listado de la *Màquina* para que puede comenzar a explorar el FORTH en su QL:

```
100 REMark Màquina FORTH
110 start up
120 :
130 REPeat forth
140 valflag=0
150 print_stack
160 AT 18,0:INK 2:PAPER 6:PRINT " OK
":PAPER 0:INK 7
170 copy_stack
180 AT 12,0:PRINT "
               "
190 REPeat keyin
200  IF INKEY$="" THEN EXIT keyin
210 END REPeat keyin
```

```
220 a$=""
230 IF doflag=0 THEN
240  AT 12,0:INPUT "? ";a$
250 ELSE
260  FOR j=pdo TO qdo-1
270   a$=a$ & s$:BEEP 100,j
280  END FOR j
290 doflag=0
300 END IF
310 position=0
320 IF a$="" THEN EXIT forth
330 a$="    " & a$ & " !!! !!! "
340 :
350 IF nword>0 AND "forget" INSTR a$"
0 AND "FORGET" INSTR a$=0 AND "list"
INSTR a$=0 AND "LIST" INSTR a$=0 AND
":" INSTR a$=0 THEN
360  j=0:p=0
370   REPeat definitions
380    j=j+1
390    IF j>nword THEN EXIT definitio
ns
400    p=LEN(w$(j))
410 :
420 FOR check=1 TO 7:BEEP 1,check
430    lencheck=LEN(a$)-p:FOR q=1 TO
LEN(a$)-p
440     IF a$(q TO q+p-1)= w$(j) THEN
450      a$=a$(1 TO q-1) & z$(j) & "
" & a$(q+p+1 TO)
460     END IF
470    END FOR q
480 END FOR check
490   END REPeat definitions
500 END IF
510 :
520 IF a$(1 TO 3)="!!!" THEN END REPe
at forth
530 strip_leading_space
540 :
550 AT 13,0
560 FOR j=1 TO 6
```

```
570 PRINT "
                    "
580 END FOR j
590 copy$=a$(1 TO LEN(a$)-9)
600 AT 12,0:PRINT "  ";copy$;"

            "
610 :
620 lena=LEN(a$)
630 REPeat action
640 :
650 search
660 strip_leading_space
670 flag=0:flag=valflag
680 IF b$(1)=":" THEN
690  define_new_word
700  EXIT action
710 END IF
720 IF position>30 THEN AT 16,0:PRINT
 "

          ":position=0
730 kk=LEN(copy$)-LEN(a$)+9:IF kk<4 T
HEN kk=4
740 IF LEN(copy$)>2 THEN
750  AT 12,2:PRINT copy$(1 TO kk-2);
760  PAPER 2:PRINT copy$(kk-1 TO ):PA
PER 0
770 END IF
780 IF b$="do" OR b$="DO" THEN flag=1
:doflag=1:DOLOOP:END REPeat forth
790 IF b$="sp!" OR b$="SP!" THEN flag
=1:empty_stack
800 IF b$="+" OR b$="-" OR b$="*"  OR
b$="/" OR b$="*/" OR b$="mod" OR b$="
MOD" OR b$="/mod" OR b$="/MOD" OR b$=
"**" THEN flag=1:operate
810 IF b$="key" OR b$="KEY" THEN flag
=1:key
820 IF b$='."' THEN flag=1:dotquote
830 IF b$="negate" OR b$="NEGATE" OR
b$="minus" OR b$="MINUS" THEN flag=1:
```

```
negate
840 IF b$="depth" OR b$="DEPTH" THEN
flag=1:depth
850 IF b$="'" THEN flag=1:tick
860 IF b$="abs" OR b$="ABS" THEN flag
=1:absolute
870 IF b$="min" OR b$="MIN" THEN flag
=1:min
880 IF b$="max" OR b$="MAX" THEN flag
=1:max
890 IF b$="swap" OR b$="SWAPH THEN fl
ag=1:swap
900 IF b$="forget" OR b$="FORGRT" THE
N flag=1:forget
910 IF b$="vlist" OR b$="VLIST" THEN
flag=1:vlist
920 IF b$="dup" OR b$="DUP" OR b$="?d
up" OR b$="?DUP" OR b$="-dup" OR b$="
-DUP" THEN flag=1:dup
930 IF b$="emit" OR b$="EMIT" THEN fl
ag=1:emit
940 IF b$="rot" OR b$="ROT" THEN flag
=1:rot
950 IF b$="spaces" OR b$="SPACES" THE
N flag=1:spaces
960 IF b$="roll" OR b$="ROLL" THEN fl
ag=1:roll
970 IF b$="over" OR b$="OVER" THEN fl
ag=1:over1
980 IF b$="drop" OR b$="DROP" THEN fl
ag=1:drop
990 IF b$="pick" OR b$="PICK" THEN fl
ag=1:pick
1000 IF b$="list" OR b$="LIST" THEN f
lag=1:list1
1010 IF LEN(b$)=1 THEN IF b$="." THEN
 flag=1:pop_stack
1020 IF NOT flag  THEN IF CODE(b$)>44
 THEN IF CODE(b$)<58 THEN number=b$:p
ush stack
1030  IF a$(1 TO 3)="!!!" THEN EXIT a
```

```
ction
1040 END REPeat action
1050 :
1060 IF a$="" THEN EXIT forth
1070 doflag=0
1080 REPeat forth
1090 :
1100 STOP
1110 :
1120 :
1130 DEFine PROCedure push_stack
1140 copy_stack
1150 :
1160 REMark Mover hacia abajo el stac
k
1170 :
1180 FOR j=19 TO 1 STEP -1
1190  n(j+1)=n(j)
1200 END FOR j
1210 :
1220 REMark poner el valor arriba
1230 n(1)=number
1240 print_stack
1250 END DEFine push_stack
1260 :
1270 DEFine PROCedure pull_stack
1280 copy_stack
1290 REMark Quitar un valor de arriba
1300 stack_val=n(1)
1310 :
1320 REMark Ahora mover el resto del
 pila una posición hacia arriba
1330 :
1340 FOR j=2 TO 20
1350  n(j-1)=n)j)
1360 END FOR j
1370 n(20)=1E-31
1380 print_stack
1390 END DEFine pull_stack
1400 :
1410 DEFine PROCedure pop_stack
1420 pull_stack
```

```
1430 AT 16,position
1440 BEEP 100,4
1450 IF stack_val=1E-31 THEN
1460  PRINT "Fila vacía":position=pos
ition + 13
1470 ELSE
1480  PRINT stack_val:position=positi
on+LEN(stack_val)+1
1490 END IF
1500 END DEFine pop_stack
1510 :
1520 DEFine PROCedure operate
1530 LOCal p,q,r,s,t
1540 p=0:q=0:r=0:s=0:t=0
1550 pull_stack
1560 p=stack_val
1570 pull_stack
1580 q=stack_val
1590 IF b$(1 TO 2)="*/" THEN
1600  pull_stack
1610  t=stack_val
1620 END IF
1630 :
1640 IF b$="+" THEN r=q+p
1650 IF b$="-" THEN r=q-p
1660 IF b$="*" THEN r=q*p
1670 IF (b$(1)="/" OR b$="mod" OR b$=
"MOD") AND p=0 THEN AT 12,0:PRINT "\*
 divisiòn por cero ilegal */":PAUSE 2
50:AT 12,0:PRINT "
                ":a$="!!!":END DEFine
operate
1680 IF b$="/" THEN r=INT(q/p)
1690 IF b$="mod" OR b$="MOD" THEN r=q
 MOD p
1700 IF b$="/mod" OR b$="/MOD" THEN s
= q MOD p:r=INT(q/p)
1710 IF b$="*/" THEN r=INT(q*t/p)
1720 IF b$="*/mod" OR b$="*/MOD" THEN
 r=INT(q*t/p):s=q*t MOD p
1730 IF b$="**" THEN r=q^p
1740 IF s= 0 THEN number=s:push_stack
```

```
1750 number=r:push_stack
1760 END DEFine operate
1770 :
1780 DEFine PROCedure search
1790  LOCal x
1800  x=0
1810 REPeat searchee
1820  x=x+1
1830  IF a$(x)=" " THEN EXIT searchee
1840 END REPeat searchee
1850 :
1860 b$=a$(1 TO x-1)
1870 IF a$(1 TO 3)="!!!" THEN END DEF
ine search
1880 a$=a$(x+1 TO)
1890 REMark
       Ahora tenemos una secciòn
       de cadena en proceso

     :
1900 END DEFine search
1910 :
1920 DEFine PROCedure empty_stack
1930 FOR j=1 TO 20
1940  n(j)=1E-31:c(j)=1E-3
1950 END FOR j
1960 END DEFine empty_stack
1970:
1980 DEFine PROCedure print_stack
1990 AT 0,0
2000 PRINT " Antes","Despuès"
2010 FOR j=1 TO 10
2020  IF c(j)=1E-31 THEN
2030   PRINT "' - '",
2040   ELSE
2050   PRINT "  ";c(j),
2060  END IF
2070  IF n(j)=1E-31 THEN
2080   PRINT "  ' - '       "
2090   ELSE
2100   PRINT "    ";n(j);"       "
2110  END IF
```

```
2120 END FOR j
2130 END DEFine print_stack
2140 :
2150 DEFine PROCedure copy_stack
2160 FOR j=1 TO 20
2170  c(j)=n(j)
2180 END FOR j
2190 END DEFine copy_stack
2200 :
2210 DEFine PROCedure swap
2220 LOCal s
2230 s=n(1)
2240 n(1)=n(2)
2250 n(2)=s
2260 END DEFine swap
2270 :
2280 DEFine PROCedure forget
2290 search
2300 FOR j=nword TO 1 STEP -1
2310  IF w$(j)=b$ THEN
2320      w$(j)=forget$
2330      z$(j)=forget$
2340      b$=""
2350  END IF
2360 END FOR j
2370 END DEFine forget
2380 :
2390 DEFine PROCedure vlist
2400 CLS
2410  FOR j=nword TO 1 STEP -1
2420    IF w$(j)<>forget$ TJEN PRINT w
$(j),
2430  END FOR j
2440 PRINT "+","-","*","/","MOD","/MO
D","*/","*/MOD","**",
2450 PRINT '."',"NEGATE","MINUS","DEP
THP,"ABS","MIN","MAXP,"SWAP","FORGET"
,
2460 PRINT "VLIST","DUP","?DUP","-DUP
","ROT","EMITP,"SPACES","ROLLP,"OVER"
,
2470 PRINT ";","DROP","PICK","LIST",
```

```
2480 PRINT "KEY","SP!","DO","LOOP"
2400 PAUSE 250:CLS:print_stack
2500 END DEFine vlist
2510 :
2520 DEFinr PROCedure dup
2530 IF (b$="?dup" OR b$="?DUP" OR b$
="-dup" OR b$="-DUP" ) AND n81)=1E-31
:AT 16,position:PRINT "Pila vacía":po
sitionlposition+12:END DEFine dup
2540 IF (b$="?dup" OR b$="?DUP" OR b$
="/dup" OR b$="-DUP"9 AND n(1)=0:AT 1
6,position:PRINT "?ero en la pila":po
sition=position+14:END DEFine dup
2550 FOR j=1 TO 19
2560  t(j)=n(j)
2570 END FOR j
2580 n(2)=n(1)
2590 FOR j13 TO 20
2600  n(j)=t(jj1)
2610 END FOR j
2620 END DEFine dup
2630 :
2640 DEFine PROCedure rot
2650 LOCal s
2660 s=n(3)
2670 n(3)=n(2)
2680 n(2)=n(19
2690 n(1)=s
2700 END DEFine rot
2710 :
2720 DEFine PROCedure emit
2730 pull_stack
2740 AT 1,position
2750 PRINT CHR$(stack_val)
2760 position=position+1
2770 END DEFine emit
2780 :
2790 DEFine PROCedure spaces
2800 pull_stack
2810 AT 16,position
2820 PRINT FILL$(" ",stack_val)
2830 position=position+stack_val
```

```
2840 END DEFine spaces
2850 :
2860 DEFine PROCedure roll
2870 LOCal s
2880 pull_stack
2890 xx=stack_val
2900 s=n(xx)
2910 FOR j=s TO 2 STEP -1
2920  n(j)=n(j-1)
2930 END FOR j
2940 n(1)=s
2950 END DEFine roll
2960 :
2970 DEFine PROCedure over1
2980 FOR j=19 TO 3 STEP -1
2990  n(j+1)=n(j)
3000 END FOR j
3010 n(3)=n(2):n(2)=n(1):n(1)=n(3)
3020 END DEFine over1
3030 :
3040 DEFine PROCedure drop
3050 FOR j=1 TO 19
3060  n(j)=n(j+1)
3070 END FOR j
3080 n(20)=0
3090 END DEFine drop
3100 :
3110 DEFine PROCedure pick
3120 LOCal x
3130 pull_stack
3140 x=stack_val
3150 FOR j=19 TO 1 STEP -1
3160  n(j+1)=n(j)
3170 END FOR j
3180 n(1)=n(x+1)
3190 END DEFine pick
3200 :
3210 DEFine PROCedure list1
3220 search
3230 IF nword=0:AT 12,0:PRINT " \* no
hay palabras definidas   */":PAUSE
50:END DEFine list1
```

```
3240 FOR j=1 TO nword
3250  IF w$(j)=b$ THEN PRINT \w$(j);"
 ",z$(j)
3260 END FOR j
3270 PAUSE 250
3280 print_stack
3290 END DEFine list1
3300 :
3310 DEFine PROCedure define_new_word
3320  BEEP 100,4
3330 nword=nword+1:IF nword>20 THEN n
word=20
3340 search
3350 w$(nword)=b$
3360 z$(nword)=""
3370 REPeat meaning
3380  search
3390  IF b$=";" THEN EXIT meaning
3400  z$(nword)=z$(nword) & " " & b$
3410 END REPeat meaning
3420 z$(nword)=z$(nword)(2 TO LEN(z$(
nword)))
3430 END DEFine define_new_word
3440 :
3450 DEFine PROCedure strip_leading_s
pace
3460  REPeat check
3470   IF a$(1)<>" " THEN EXIT check
3480   a$=a$(2 TO)
3490  END REPeat check
3500 END DEFine strip_leading_space
3510 :
3520 DEFine PROCedure negate
3530  pull_stack
3540  number=-stack_val
3550  push_stack
3560 END DEFine negate
3570 :
3580 DEFine PROCedure absolute
3590  pull_stack
3600  number=ABS(satck_val)
```

```
3610  push_stack
3620 END DEFine absolute
3620 :
3640 DEFine PROCedure min
3650  LOCal p,q
3660  pull_stack
3670  p=stack_val
3680  pull_stack
3690  q=stack_val
3700   IF q<p THEN
3710     number=q
3720   ELSE
3730     number=p
3740   END IF
3750  push_stack
3760 END DEFine min
3770 :
3780 DEFine PROCedure max
3790  LOCal p,q
3800  pull_stack
3810  p=stack_val
3820  pull_stack
3830  q=stack_val
3840   IF q>p THEN
3850     number=q
3860   ELSE
3870     number=p
3880   END IF
3890  push_stack
3900 END DEFine max
3910 :
3920 DEFine PROCedure depth
3930 LOCal m:m=0
3940 REPeat count
3950  m=m+1
3960  IF n(m)=1E-31 THEN number=m-1:E
XIT count
3970 END REPeat count
3980 push_stack
3990 END DEFine depth
4000 :
4010 DEFine PROCedure key
```

```
4020  key$=""
4030 REPeat clearing
4040  IF INKEY$="" THEN EXIT clearing
4050 END REPeat clearing
4060 REPeat get_key_press
4070  key$=INKEY$
4080  IF key$<>"" THEN EXIT get_key_p
ress
4090 END REPeat get_key_press
4100  number=CODE(key$)
4110  push_stack
4120 END DEFine key
4130 :
4140 DEFine PROCedure dotquote
4150 AT 16,position
4160 REPeat printing
4170  search
4180  IF b$(LEN(b$))='"' THEN EXIT pr
inting
4190  PRINT b$;" ";
4200  position=position+LEN(b$)+1
4210 END REPeat printing
4220 PRINT b$(1 TO LEN(b$)-1)
4230 position=position+LEN(b$)-1
4240 END DEFine dotquote
4250 :
4260 DEFine PROCedure tick
4270 search
4280 yesflag=0
4290 FOR j=1 TO nword
4300  IF w$(j)=b$ THEN yesflag=1
4310 END FOR j
4320 IF yesflag=1 THEN
4330  AT 16,position
4340  PRINT b$;" OK"
4350 END IF
4360 END DEFine tick
4370 :
4380 DEFine PROCedure doloop
4390 s$=""
4400 pull_stack
4410 pdo=stack-val
```

```
4420 pull_stack
4430 qdo=stack_val
4440 IF pdo>=qdo THEN AT 16,position:
PRINT "Valores DO LOOP ilegales":dofl
ag=0:END DEFine doloop
4450 REPeat loop_tasks
4460  IF b$="loop" OR b$="LOOP" THEN
EXIT loop_tasks
4470  s$=s$ & " " & b$
4480  search
4490 END REPeat loop_tasks
4500 s$=s$(5 TO) & " "
4510 END DEFine doloop
4520 :
4530 DEFine PROCedure start_up
4540 CLS:CLS #0
4550 AT 0,0
4560 PRINT "Pulse 1 para arranque en
caliente           2 para arranque en
frio"
4570 REPeat yeti
4580  IF INKEY$="" THEN EXIT yeti
4590 END REPeat yeti
4600 REPeat yeti2
4610  k$=INKEY$
4620  IF k$="1" OR k$="2" THEN EXIT y
eti2
4630 END REPeat yeti2
4640 CLS
4650 IF k$="2" THEN initialise
4660 END DEFine start_up
4670 :
4680 :
4690 DEFine PROCedure initialise
4700 PAPER 2:INK 6:BORDER 0
4710 CSIZE 3,1
4720 CLS:CLS #0
4730 FLASH 1:AT 5,1:PRINT "Inicializa
ndo MAQUINA FORTH":FLASH 0
4740 CSIZE 0,0
4750 REMark n - números
4760 REMark c - copiar la pila
```

```
4770 REMark t - almac. temporal
4780 REMark w$ - nombre palabra nueva
4790 REMark z$ - signif. palab. nueva
4800 DIM n(20),c(20),t(20),w$(20,400)
,z$(20,400)
4810 stack=0
4820 a$="":b$=""
4830 number=0:stack_val=0:s=0:doflag=
0
4840 nword=0:REMark contador palabras
                          nuevas
4850 forget$=FILL$("*",10)
4860 FOR j=1 TO 20
4870  w$(j)=forget$
4880  z$(j)=forget$
4890 END FOR j
4900 REPeat clearing
4910  IF INKEY$="" THEN EXIT clearing
4920  BEEP 100,j:PAUSE 1
4930 END REPeat clearing
4940 PAPER 0:INK 7:CLS
4950 empty_stack
4960 END DEFine initialise
4970 :
```

Si tiene interès en seguir investigando el FORTH, aquî le doy unos libros que pueden serle ûtiles:

BASIC and FORTH in Parallel - S J Wainwright (Babani, 0 85934 113 5, ±1.95)

Discover FORTH - Thom Hogan (McGraw-Hill, 0 931988 79 9, ±11.25).

Starting FORTH - Leo Brodie (Prentice-Hall, 0 13 8429227, ±18.40. Este es probablemente el mejor libro de FORTH para los principiantes).

The complete FORTH - Alan Winfield (Sigma, 0 905 104226, ±6.95).

Exploring FORTH - Owen Bishop (Granada, 0 246 121882, £6.95).

Learn FORTH on your BASIC Micro - Tim Hartnell y Paul Kail (Interface Publications =td., 1984, £5.95).

FORTH es una marca registrada de FORTH, Inc., 2309 Pacific Coast Highway, Hermosa Beach, California, 90254, USA.

Capítulo Ventidos

Simulando la Realidad

La simulaciòn es una forma de fingir una realidad. A diferencia de las aventuras (de las que trataremos en el proximo capìtulo) donde la 'realidad' creada por el ordenador suele ser màgica o del tipo de los sueños, el mundo al que se accede a travès de la simulaciòn està generalmente mas cerca del suelo.

En una simulaciòn, el ordenador manipula variables de acuerdo con las formulas que se le especifican, manteniendo las reglas de la situaciòn que se ha generado y tomando el lugar del simulado desde el punto de vista de las reaciones a los datos que se le suministran.

Las simulaciones intentan hacer una rèplica de la vida. Sin embargo, como la vida es demasiado complicada como para tener unos lìmites, necesitamos de cierta simplificaciòn antes de que podamos producir una situaciòn simulable. A pesar de esta simplificaciòn, como verà pronto, una simulaciòn bien desarrollada puede imitar la vida con bastante aproximaciòn. Las causa y los efectos estàn tan intimamente ligados que no es demasiado complicado preparar una formula que los trate. El generador de nùmeros aleatorios del QL puede tomar el lugar de cambios imprevistos como el tiempo, cambios demogràficos, el comportamiento de las molèculas de un gas o la forma en que se dispersa la tinta en un medio lìquido.

En este capìtulo veremos solamente una simulaciòn simple. Sin embargo, el programa es bastante grande y al mismo tiempo plantea cierto nùmero de problemas a resolver e indica las formas en que usted puede desarrollar sus propias simulaciones en el QL.

El programa de simulaciòn es *Director de Operaciones*, en que usted tiene que hacer funcionar una factorìa que

produce una gran variedad de artículos (pero solamente un tipo de objetos cada vez que el programa es ejecutado) desde bicicletas hasta ZX Spectrum y QLs. Usted es el director de la factoría. Tiene un control casi total del personal, sus productos y el precio de venta. Su tarea consiste en permanecer con el negocio en marcha tanto tiempo como le sea posible, o hasta que haya acumulado 10.000 dòlares.

En este programa, que es autoexplicativo, usted està encargado de la factoría. Al principio le dice cuanto personal tiene en plantilla, cuànto se le paga cada semana, el capital que tiene disponible, las exixtencias del almacèn y el precio de venta de los objetos que se manufacturan. El programa le dice a cuanto ascienden sus deudas, èste es un concepto que debe tener siempre presente. El programa trabaja por semanas y su gestiòn debe tender a compensar sus ingresos y sus gastos o caerà en la bancarrota.

Bancarrota

El programa puede terminar de dos formas. La màs usual es la bancarrota, en la que le dice cuantas semanas a dirigido la factoría satisfactoriamente. La segunda forma es alcanzando un capital de 10.000 dòlares. Como descubrirà a medida que practique esta simulaciòn, las cartas estàn contra usted. Hay un dicho en el mundo de los negocios que dice que lo primero que se debe hacer en un negocio es asegurar su supervivencia. Usted descubrirà còmo lograrlo cuando intente hacer funcionar su factoría en el QL. A pesar de la gran simplificaciòn que se ha hecho para que pueda funcionar esta simulaciòn, la rèplica de la realidad es bastante aproximada. Verà còmo se desespera cuando algunos artículos son devueltos y còmo tiene que modificar sus decisiones sobre los precios de mercado de ciertos artículos.

El programa le parecerà bastante confuso cuando lo introduzca pero, no se asuste aunque tenga que hacer frente a un gran nùmero de decisiones, el QL guiarà sus pasos

facilitandoselas.

Ahora es el momento de introducir el programa y experimentar por sí mismo las delicias de ser un *Director de Operaciones*:

```
10 REMark   DIRECTOR DE OPERACIONES
20      inicializar
30 REPeat ciclo_juego
40      we=we+1
50      informe
60      plantilla
70      informe
80      produccion
90      informe
100     ventas
110     problemas
120     ca=ca-wa*wo-rc
130 END REPeat ciclo_juego
140 REMark -------------
150 DEFine PROCedure problemas
160 CLS
170 IF RND>.45 THEN
180  a=RND(1 TO 7)
190  PRINT \\\"   Los sindicatos pide
n una"
200  PRINT "     subida del ";a"%"
210  wa=INT(100*(wa+wa*a/100))/100
220  espera
230  PRINT \" Sueldo actual por emple
ado ";wa;"$"
240  espera
250  CLS
260 END IF
270 IF RND>suceso THEN
280  suceso=.81
290  PRINT\\\"  Un fuego a destruido
parte de"
300  PRINT "  sus existencias del alm
acèn."
310  PRINT "  en breve recibirà un in
forme de"
```

```
320  PRINT "  los daños causados...
330  espera
340  a=INT(RND*st/2)+1
350  st=st-a
360  PRINT \"  Se han destruido ";a;o
bjetos$
370  PRINT "  Estaban valorados en ";
a*sp;"$"
380  espera
390  PRINT " Las existencias son ";st
;objetos$
400  espera
410 END IF
420 IF RND<.3 THEN
430  CLS
440  PRINT \\\" Su principal proveedo
r ha anunciado"
450  PRINT "   una dramàtica subida d
e precios.."
460  espera
470  a=INT(RND*100*co/7)/100
480  IF a<1E-2 THEN GO TO 470
490  PRINT \" El costo de fabricaciòn
 de ";objetos$
500  PRINT "   ha subido en ";a;"$ po
r unidad"
510  espera
520  co=co+a
530  PRINT \"  Ahora cuesta ";co;"$"
540  PRINT "        cada unidad"
550  espera
560 END IF
570 IF RND<.65 AND ma<sp THEN END DEF
ine problemas
580 CLS
590 PRINT \\\" Tiene la oportunidad d
e elevar los"
600 PRINT "   precios. Sus";objetos$
610 PRINT "   se venden ahora a ";sp;
"$"
620  espera
630 PRINT\" Que porcentaje de increme
```

```
nto"
640 PRINT "     va a imponer?
                              ";
650 INPUT a
660 re=re+a
670 sp=INT(100*(sp+a*sp/100))/100
680  espera
690 PRINT \\"EL ";objetos$;" se vende
 ahora a ";sp;"$"
700  espera
710 END DEFine problemas
720 REMark ----------------
730 DEFine PROCedure ventas
740  PRINT \"  Sus existencias de ";o
bjetos$;
750  PRINT "           es ahora de ";st
760    espera
770  FLASH 1:PAPER 0:INK 7
780  PRINT \\" Van a pasarle un infor
me de ventas"
790    espera
800    FOR slide=1 TO 24
810     PRINT "
                 "
820    END FOR slide
830  FLASH 0
840    FOR slide=1 TO 18
850     PAPER RND(1 TO 7):AT slide,0
860     PRINT "
                 "
870    PAUSE 4
880    END FOR slide
890  PAPER 2:INK 6
900  IF a>st THEN GO TO 830
910  CLS
920  PRINT\\\" El nùmero total de ";o
bjetos$
930  PRINT "      vendidos fuè ";a
940  st=st-a
950  za=a*sp
960  PRINT \"  Los ingresos fueron ";
za;"$"
```

```
970 ca=INT(a*sp*100)/100+ca
980   espera:PAUSE 100
990 END DEFine ventas
1000 REMark ----------------
1010 DEFine PROCedure informe
1020 CLS
1030  IF cas+st<1 THEn bancarrota:STO
P
1040  IF ca+st>9999 THEN exito:STOP
1050   PAPER 7:INK 2
1060  PRINT \" Informe de tienda, sem
ana ";we;" "
1070   PAPER 2:INK 7
1080  PRINT \" Capital en caja ";INT(
ca*100)/100;"$"
1090  PRINT " Gastos generales ";rc;"
$ por semana"
1100  PRINT \" En almacèn hay ";st;ob
jetos$
1110   PRINT "   valorados en ";INT(s
t*sp*100)/100;"$"
1120  PRINT \" Se vendieron a ";sp;"$
 cada uno, y"
1130  PRINT "          costaron ";co;"$
 unidad"
1140  espera
1150  PRINT \"  Su plantilla es de ";
wo;", "
1160  PRINT "   y les paga ";wa;"$ a
cada uno"
1170  PRINT "  por semana. La nòmina
de "
1180  PRINT "        esta semana es de
";wa*wo;"$"
1190   espera
1200  PRINT \"  Cada persona fabrica
";pr
1210  PRINT "   ";objetos$;", un tota
l de ";pr*wo
1220 END DEFine informe
1230 REMark --------------
1240 DEFine PROCedure produccion
```

```
1250  PRINT \" Que cantidad quiere pr
oducir?                      "
1260   INPUT ma
1270   IF ma=0 THEN END DEFine produc
cion
1280   PRINT
1290    IF ma*co>ca THEN
1300     PRINT " No tiene suficiente
capital"
1310    espera:PRINT "              ";
1320    GO TO 1260
1330   END IF
1340   IF ma>pr*wo THEN
1350    PRINT "  No tiene suficiente
gente en              su plantilla"
1360    espera
1370    PRINT "                    ";
1380    GO TO 1260
1390   END IF
1400  PRINT "Si Sr. el objetivo de la
 semana ";we
1410  PRINT "              es ";ma;objeto
s$
1420  ma=ma-INT(RND*ma/5+1)
1430  espera
1440  PRINT \"    El nùmero de ;"obje
tos$
1450  PRINT "  producidos la semana "
;we
1460  PRINT "                 fuè ";ma
1470  st=st+ma
1480  ca=ca-co*ma
1490  espera
1500 END DEFine produccion
1510 REMark -----------------
1520 DEFine PROCedure plantilla
1530  a=0
1540  PRINT \"  Cuanta gente quiere"
1550  PRINT "                contratar?"
1560  INPUT a
1570  IF a=0 THEN GO TO 1630
1580  wo=wo+a
```

```
1590  PRINT \"  La plantilla es ahora
 de ";wo
1600  espera
1610 END DEFine plantilla
1620   informe
1630  PRINT \"A cuanta gente quiere d
espedir?                          ";
1640  INPUT a
1650  IF a=0 THEN
1660   espera
1670   END DEFine plantilla
1680  END IF
1690  IF a>wo THEN GO TO 1640
1700  a=RND(1 TO a)
1710  espera
1720  PRINT \"Los sindicatos le permi
ten despedir              a ";a
1730  wo=wo-a
1740  espera
1750 END DEFine plantilla
1760 REMark ----------------
1770 DEFine PROCedure bancarrota
1780  PRINT \"        Està en bancarro
ta!!"
1790  espera
1800  PRINT \"        Que bochorno!"
1810  espera
1820  PRINT \"   Ha conseguido manteb
nerse             en el negocio ";we
;" semanas"
1830 END DEFine bancarrota
1840 REMark ------------
1850 DEFine PROCedure inicializar
1860  RANDOMISE
1870  RESTORE
1880  INK 7:PAPER 2:BORDER 1,6
1890   FOR z=1 TO RND(1 TO 10)
1910    READ objetos$
1920   END FOR z
1930  suceso=.61
1940  ca=RND(500 TO 999)
```

```
1950  st=RND(100 TO 599)
1960  sp=RND(10 TO 14)
1970  co=RND(7 TO 11)
1980  IF co>sp THEN GO TO 1970
1990  wo=RND(7 TO 16)
2000  wa=12+INT(RND*sp)
2010  pr=RND(5 TO 10)
2020  rc=RND(100 TO 119)
2030  we=0
2040  re=1:
REMark re = 'factor de resistencia a
                        las ventas'
2050 END DEFine inicializar
2060 REMark ---------------
2070 DEFine PROCedure espera
2080  BEEP 32000,RND(40 TO 60),RND(1
TO 5),RND(1 TO 5),RND(1 TO 5),RND(1 T
O 5),RND(1 TO 5),-22286
2090  FOR espera_por=1 TO 700
2100  END FOR espera_por
2110 END DEFine espera
2120 REMark ---------------
2130 DEFine PROCedure exito
2140  PRINT \\\" Ya ha ganado 10.000$
 y ya"
2150  PRINT "        puede retirarse"
2160 END DEFine exito
2170 REMark ---------------
2180 DATA " Sinclair QLs"," bicicleta
s"," armónicas"," televisores"
2190 DATA " sillas "," cocinas"," coc
hes"
2200 DATA " zx Spectrums"," abrigos"
```

Capítulo Ventitres - Creando Aventuras

Seamos realistas, la vida es bastante tranquila algunas veces. No parece que haya en mi ciudad muchos dragones esperando en las esquinas para devorarme, ni castillos abandonados con tesoros esperando a un valiente que vaya a rescatarlos. Ya he olvidado la última vez que me topè con un Diabòlico Mago en la cola del supermercado y ha pasado mucho tiempo desde la última vez que discutí sobre tàcticas de guerra galàctica con los androides en la taberna de la esquina.

La capacidad de fantasía està dentro de cada uno de nosotros. El deseo de tomar la personalidad de otro, màs vibrante, aùnque solo sea por unas horas, es algo muy comùn. Sin embargo usted no puede conjurar diablos y monstruos abominables, invocar el poder de la Coraza de Protecciòn o emplear gnomos para llevar sacos de esmeraldas desde las ruinas de un castillo abandonado. Los juegos de aventuras le permiten hacerlo.

Probablemente habrà visto màs de un anuncio que dice que es el 'juego de mayor aceptaciòn en el mundo'. Independientemente de que sea cierto o no, esto indica que los juegos de aventuras son una forma de emplear el ocio que satisface las necesidades internas de una gran cantidad de gente.

Usted tambièn habrà tomado parte alguna vez en un juego de aventuras. Pero estas campañas tienen una gran desventaja, necesita gente que juegue con o contra usted y no es normal que encuentre suficiente gente dispuesta a hacerlo cuando usted lo desea. Aquí es donde el QL entra en acciòn.

Imprevisible

A pesar que los juegos de aventuras del QL adolecen de

cierta falta de expontaneidad que necesitan ciertos juegos cuando se juegan con compañía humana, pueden ser suficientemente imprevisibles y excitantes de jugar. El hecho de que el temible Monstruo del Tio Clive solamente exista dentro de la RAM del QL, no le quita valor a la satisfacción que se siente al matarlo. Las gemas que encuentra en el lugar no son tan irreales como las que se descubren en aventuras de acción.

La palabra aventura se usa para describir la clase de juegos de ordenador en la que el jugador se mueve a travès de una realidad alternativa. En este mundo hay monstruos con los que luchar, tesoros que descubrir, mapas que construir y rompecabezas que resolver.

Hay dos tipos de programas de aventuras en este capítulo, ambos son largos y deben hacerle trabajar durante horas para resolverlos.

Una característica de los juegos de aventuras es que la realidad que imitan es consistente. Esto es, el mundo que se crea dentro del programa es sòlido y -aparte de ciertos sucesos específicos, como un estremecimiento o un conjuro màgico- las partes del mundo no pasan de forma aleatoria. En una aventura bien construida los rios permanecen en su lugar, las paredes de las mazmorras no se mueven misteriosamente cuando las vuelve la espalda y los objetos que usted pone en una caverna dentro de un laberinto subterràneo, no aparecen o desaparecen de repente.

Construir los mapas es una de las partes màs divertidas de los juegos de aventuras. Preparando el camino a travès de un entorno de monstruos y recogiendo tesoros, resolviendo rompecabezas a medida que sube y baja escaleras, explorando tùneles, perdiendose en laberíntos, etc. , sòlo es entretenido si se puede trazar un mapa de ello. Los dos "mundos" creados en este capítulo son susceptibles de ser representados en un mapa y, el primero de ellos, le muestra de vez en cuando un mapa visto desde arriba.

Nuestro primer programa es El Marquès de Kwe-El. Esta es una "aventura de parrilla". En otras palabras, el "mapa del

mundo" se basa en una parrilla, y usted se mueve hacia arriba(norte), hacia abajo(sur), a la derecha(este) y la izquierda(oeste) de la parrilla. Este mundo està plagado de trampas temibles, incluyendo pozos con arenas movedizas o dragones, asì como cuevas màgicas que le pueden transportar a lugares aleatorios dentro de la tierra de Kwue-EL. De vez en cuando, como ya le dije, se le mostrarà ràpidamente un trozo del mapa. En èl, las interrogaciones son cuevas, el signo del dòlar son tesoros, los bloques sòlidos son barreras infranqueables ... y asì sucesivamente. (Ya le explicarè el significado de los otros sìmbolos).

Si es usted un veterano en los juegos de ordenador, reconocerà que este programa se ha desarrollado partiendo del viejo juego del *Cazador de Vampiros*. En este programa, a diferencia del original *Cazador*, puede luchar con el dragòn si lo desea. Al principio del juego dispone de cinco flechas, y puede disparar dentro de cualquier caverna en una de las cuatro direcciones.

Necesita sobrevivir durante 25 minutos para ganar el juego, y su puntuaciòn final depende de a cuantos dragones haya vencido y cuantos tesoros haya encontrado. Usted tiene un amuleto màgico que le proporciona valiosa informaciòn acerca del contenido de las cavernas que se encuentran cerca de usted. El èxito en este juego depende de su ingenio y habilidad para visualizar el sistema de cavernas.

Cuando estè preparado para seguir las indicaciones y arriesgar su vida, introduzca y ejecute este programa:

```
10 REMark El Marquès de Kwue-El
20   inicializar
30   CLS:PRINT \\\
40   mapa
50 REPeat ciclo_juego
60   q=RND(0 TO 6)
70   IF q=0 AND e<>55 THEN map
80  INK 6:PAPER 2
90   CLS:PRINT\\\\
```

```
100   PRINT " Marquês ";nombre$;", es
tâs en la cueva ";e
110   IF g>0 THEN PRINT " Llevas ";:I
NK 0:PAPER 7:FLASH 1:PRINT g;"$";:INK
 6:PAPER 2:FLASH 0:PRINT " de oro"
120   amuleto
130 PRINT \"  Te quedan ";:PAPER 6:IN
K 2:PRINT 25-h;:PAPER 2:INK 6:PRINT "
 unidades de carisma"
140     FLASH 1
150 PRINT \"   Que deseas hacer?"
160     FLASH 0
170 IF INKEY$<>"" THEN GO TO 170
180 PRINT \"N - ir al norte    S - ir
 al sur"
190 PRINT "E - ir al este     O - is
al oeste"
200 PRINT "L - luchar         Q - aba
ndonar"
210 j$=INKEY$:BEEP 100,RND(3 TO 13)
220 IF j$<>"n" AND j$<>"N" AND j$<>"s
" AND j$<>"S" AND j$<>"e" AND j$<>"E"
 AND j$<>"o" AND j$<>"O" AND j$<>"l"
AND j$<>"L" AND j$<>"q" AND j$<>"Q" T
HEN GO TO 210
230 u=0
240 IF (j$="n" OR j$="N") AND a(e-10)
=88 OR (j$="s" OR j$="S") AND a(e+10)
=88 OR (j$="e" OR j$="E") AND a(e+1)=
88 OR (j$="o" OR j$="O") AND a(e-!)=8
8 THEN
250  PRINT "No te puedes mover en esa
 direcciòn!"
260  GO TO 210
270 END IF
280 IF j$="q" OR j$="Q" THEN
290  q=9
300  fin_del_juego
310  STOP
320 END IF
330 a(e)=46
340 IF j$="n" OR j$="N" THEN e=e-10
```

```
350 IF j$="s" OR j$="S" THEN e=e+10
360 IF j$="e" OR j$="E" THEN e=e+1
370 IF j$="o" OR j$="O" THEN e=e-1
380 IF j$="l" OR j$="L" THEN lucha_dr
agon
390 IF a(e)=63 THEN magico
400 IF a(e)=68 THEN dragon
410 IF a(e)=81 THEN arenas
420 IF a(e)=36 THEN oro
430 h=h+1
440 IF h=25 THEN
450   q=9
460   fin_del_juego
470   STOP
480 END IF
490 espera
500 END REPeat ciclo_juego
510 REMark --------------
520 DEFine PROCedure magico
530  INK 7:FLASH 1
540 PRINT \" Marquès ";nombre$;", has
 caido "
550 PRINT " en una cueva màgica, y ah
ora serås"
560 PRINT " arrastrado a otra cueva..
."
570  INK 6:FLASH 0
580  FOR cambridge=1 TO 6
590   FOR busqueda=1 TO 6
600    BEEP 10900,busqueda,199,1,1
610    PAUSE 10/busqueda:BEEP 1000,ca
mbridge
620    BEEP 10090,10-cambridge,224,15
,1,11,7
630    PAUSE 15/cambridge
640   END FOR busqueda
650  END FOR cambridge
660 a(e)=46
670 REPeat arrastre
680  e=RND(12 TO 87)
690  IF a(e)<>88 THEN EXIT arrastre
700 END REPeat arrastre
```

```
710 END DEFine magico
720 REMark ---------------
730 DEFine PROCedure dragon
740 PRINT \\"Estás en la guarida del
dragón..."
750 PRINT "      Comienza a rezar tus
 oraciones"
760 espera
770  INK 5:PRINT \"------------------
----------------":INK 6
780 m=RND(1 TO 5)
790 IF m=1 THEN
800  PRINT "El dragón ha huido"
810  espera
820  END DEFine dragon
830 END IF
840 PRINT "Se ha despertado...y te ha
 visto!"
850 espera
860 IF m=5 THEN
870  PRINT "Hace poco que ha comido"
880  PRINT "y se vuelve a dormir"
890  espera
900  END DEFine dragon
910 END IF
920 PRINT \"Ahora te va a atacar"
930 espera
940 m=RND(1 TO 10)
950 IF m>8 THEN
960  PRINT "pero tu luchas valienteme
nte..."
970  espera
980  PRINT "       ...y le ganas"
990  END DEFine dragon
1000 END IF
1010 PRINT "Adiós Marqués, ";nombre$
1020   espera
1030   q=9
1040   fin_del_juego
1050   STOP
1060 END DEFine dragon
1070 REMark ---------------
```

```
1080 DEFine PROCedure arenas
1090  FOR flap=1 TO 10
1100   MODE 4:PAUSE 2:MODE 8:PAUSE 2:
BEEP 1000,flap*flap
1110  END FOR flap
1120 FOR suck=1 TO 12
1130  FOR fango=1 TO suck
1140   PRINT " ";
1150   BEEP 9600,suck*fango
1160  END FOR fango
1170   tinta=RND(0 TO 7):papel=7-tint
a
1180   STRIP papel:PRINT "Horror....a
renas movedizas"
1190  BEEP 19700,27,95,14,suck,10,8,2
0770:PAUSE 5
1200 END FOR suck
1210 espera:espera
1220   PAPER 2:INK 6
1230 PRINT "

              "
1240   q=9:h=0
1250   fin_del_juego
1260   STOP
1270 END DEFine arenas
1280 REMark ------------------
1290 DEFine PROCedure oro
1300 k=RND(100 TO 199)
1310 PRINT
1320 FOR gemas=1 TO 12
1330 PRINT "  ";
1340     FOR brillo=1 TO gemas
1350      PRINT "$";
1360      BEEP 9950,brillo*gemas:PAUSE
 4:BEEP 9950,(brillo+1)/gemas
1370     END FOR brillo
1380    INK RND(4 TO 7)
1390    PRINT "  El tesoro..."
1400    PRINT
1410    BEEP RND(3200 TO 23200),gemas/
brillo,1,11,-6,11,7,871
```

```
1420   PAUSE gemas
1430 END FOR gemas
1440 espera
1450 INK 7
1460 PRINT "Has encontrado el tesoro
del dragòn"
1470 PRINT "     oron valorado en ";k
;"$"
1480 g=g+k
1490 espera
1500 END DEFine oro
1510 REMark ---------------
1520 DEFine PROCedure amuleto
1530 y=1
1540 REPeat busca
1550  l=a(e+p(y))
1560  IF l<>46 OR y=8 THEN EXIT busca
1570  y=y+1
1580 END REPeat busca
1590 IF l=46 THEN END DEFine amuleto
1600 PRINT \"  Tu amuleto señala que"
1610 PRINT " hay ";
1620 SELect ON l
1630  ON l=88
1640   PRINT "una pared sòlida"
1650  ON l=63
1660   PRINT "una cueva màgica"
1670  ON l=68
1680   PRINT "un dragòn"
1690  ON l=81
1700   PRINT "arenas movedizas"
1710  ON l=36
1720   PRINT "oro "
1730 END SELect
1740 PRINT "muy cerca"
1750 espera
1760 END DEFine amuleto
1770 REMark ---------------
1780 DEFine PROCedure lucha_dragon
1790 INK 7:PAPER 1
1800 ar=ar-1
1810   IF ar=0 THEN
```

```
1820    PRINT "Has usado todas tus fl
echas"
1830    espera
1840    END DEFine lucha_dragon
1850   END IF
1860 PRINT \\"Te quedan ";ar;" flecha
s"
1870 ss=0
1880 PRINT "En que direcciòn quieres
disparar    (N, S, E u O), ";nombre$;
"?"
1890 REPeat disparo
1900  s$=INKEY$
1910  IF (s$="n" OR s$="N") AND a(e-1
0)=68 THEN ss=1:yt=e-10
1920  IF (s$="s" OR s$="S") AND a(e+1
0)=68 THEN ss=1:yt=e+10
1930  IF (s$="e" OR s$="E") AND a(e+1
)=68 THEN ss=1:yt=e+1
1940  IF (s$="o" OR s$="O") AND a(e-1
)=68 THEN ss=1:yt=e-1
1950  IF s$="n" OR s$="N" OR s$="s" O
R s$="S" OR s$="e" OR s$="E" OR s$="o
" OR s$="O" THEN EXIT disparo
1960 END REPeat disparo
1970 PRINT
1980 IF ss=0 THEN
1990  PRINT "Na està el dragòn ahì...
"
2000  espera
2010  PRINT "Has malgastado una flech
a"
2020  espera
2030  END DEFine lucha_dragon
2040 END IF
2050 PRINT \"Muy bien, Marquès ";nomb
re$
2060 PRINT \"Has acertado al feroz
 dragòn"
2070 espera
2080 IF RND(1 TO 3)=2 THEN
2090  PRINT "Lo has matado!"
```

```
2100   a(yt)=46:k=RND(100 TO 200)
2110  PRINT \"Seràs recompensado con
";k;"$"
2120  g=g+k
2130 ELSE
2140 PRINT "Pero solo està herido..."

2150 END IF
2160 espera
2170 END DEFine lucha_dragon
2180 REMark ---------------
2190 DEFine PROCedure fin_del_juego
2200 PAPER 2:INK 6
2210 IF h<1 THEN
2220  PRINT "Se te ha acabado todo tu
 carisma..."
2230  ELSE
2240   PRINT "Te quedan ";25-h;" unid
ades de carisma"
2250 END IF
2260 espera
2270 IF g>0 THEN
2280  PRINT "Has conseguido ";g;"$ en
 oro"
2290 END IF
2300   mapa
2310 END DEFine fin_del_juego
2320 CSIZE 3,0
2330 DEFine PROCedure mapa
2340 PRINT \\
2350 CSIZE 1,0
2360 a(e)=72
2370 PRINT "  ";
2380 FOR j=1 TO 100
2390  PAPER 2:INK 6
2400  m=a(j)
2410  IF a(j)=88 THEN INK 1:PAPER 7
2420  IF a(j)=63 THEN INK 7:PAPER 3:F
LASH 1
2430  IF a(j)=68 THEN INK 6:PAPER 2
2440  IF a(j)=81 THEN INK 7:PAPER 1
```

```
2450  IF a(j)=36 THEN INK 2:PAPER 7
2460  IF a(j)=72 THEN INK 0:PAPER 5
2470  PRINT CHR$(m);
2480  PAPER 2:INK 6:FLASH 0
2490  PRINT " ";
2500  IF 10*INT(j/10)=j THEN PRINT:PR
INT "  ";
2510 END FORj
2520   espera
2530 CSIZE 0,0
2540 END DEFine mapa
2550 REMark ----------------
2560 DEFine PROCedure inicializar
2570 BORDER 3,1
2580 PAPER 2:INK 6
2590 CLS:CLS #0
2600 PRINT \\"  Bienvenido al mundo d
e Kwue-El"
2610   espera:espera
2620 PRINT \" Tu misiòn es explorar l
as"
2630 PRINT " cuevas de este mundo, en
 busca del"
2640 PRINT " tesoro, e intentar captu
rar al"
2650 PRINT " malèfico dragòn que vive
 en ellas"
2660 PRINT " Cuàl es tu nombre?"
2670 PRINT \\\"               ";
2680 INPUT nombre$
2690  nombre$=nombre$(1 TO 7)
2700 PAPER 6:INK 0
2710 CLS
2720 PRINT \\\\"    Saludos, Marquès
";nombre$
2730  espera
2740 PRINT \\"  Empiezas la exploraci
òn con"
2750 PRINT "   25 unidades de carisma
, y debes"
2760 PRINT "   terminar tu tarea ante
s que tu"
```

```
2770 PRINT "   carisma se haya agotad
o."
2780 PRINT "   consumiràs una unidad
por cada"
2790 PRINT "   movimiento que hagas..
"
2800 DIM a(100):DIM p(8)
2810 h=0:q=0:l=0:g=0:ar=6
2820 FOR b=1 TO 100
2830  a(b)=46
2840  BEEP 900,b
2850  IF b<12 OR b>90 OR 10*INT(b/10)
=b OR 10*INT(b/10)=b-1 THEN a(b)=88
2860 END FOR b
2870 FOR b=1 TO 5
2880    RESTORE
2890   FOR d=1 TO 5
2900    z=RND(12 TO 87)
2910     IF a(z)=88 THEN GO TO 2900
2920    READ a(z)
2930    BEEP 1000,d*b
2940   END FOR d
2950  END FOR b
2960 FOR b=1 TO 8
2970  READ p(b)
2980 END FOR b
2990 e=55
3000  PAPER 2:INK 6:CLS
3010 END DEFine inicializar
3020 REMark -----------------
3030 DEFine PROCedure espera
3040  FOR paciencia=1 TO 10
3050   BEEP 1000,RND(1 TO 100),RND(10
0 TO 200),RND(1 TO 8),RND(1 TO 8),RND
(1 TO 32000)
3060   PAUSE 4
3070  ENF FOR paciencia
3080 END DEFine espera
3090 REMark ----------------
3100 DATA 88,63,68,81,36
3110 DATA -11,-10,-9,-1,1,9,10,11
```

Los Lobos Fantasmas

Nuestra segunda aventura se llama *Los Lobos Fantasmas y el Vagabundo* (donde usted es el vagabundo y los lobos fantasmas y otras criaturas terribles habitan el entorno que va usted a visitar), y tiene lugar dentro de un viejo y abandonado castillo de piedra.

Usted ha leìdo acerca del castillo en una vieja carta que encontrò en un baùl que le legò su abuelo. Desafortunadamente, solo ha podido encontrar la segunda pàgina de la carta, de modo que no està seguro de la historia completa. Sin embargo, por la pàgina que queda, ha entendido que el castillo fuè abandonado hace siglos, despuès de que una vieja bruja lanzara un conjuro contra sus habitantes durante una terrible tormenta. La esposa del rey enfermò y èl, equivocadamente, culpò a la bruja de la enfermedad de su esposa. Pensò que echando lejos a la bruja, cesarìa la maligna influencia que ejercìa sobre su esposa. Esto no sucediò, y su mujer empeorò hasta la muerte.

Sus ùltimos dias no fueron pacìficos. La maldiciòn de la vieja bruja convirtiò el castillo en un reino de terror ya que terribles criaturas y fantasmas aparecieron en èl. Finalmente, el rey y su corte no pudieron permanecer allì por màs tiempo y huyeron de su hogar sin que se volviera a saber de ellos nunca màs.

Las criaturas invocadas por la bruja permanecieron y aùn viven hoy allì, y usted va a entrar ahora en sus dominios.

El programa comienza preguntàndole su nombre, para poder usarlo a lo largo de toda la aventura. El QL le dirà a continuaciòn -asì como al iniciar cualquier acciòn- su estado actual. Le dirà cuànto dinero tiene, el estado de su fuerza y otros detalles como, cuàndo puede llevar la armadura.

A continuaciòn le preguntarà què quiere hacer, y usted le darà instrucciones con letras simples. Necesita empezar con una "I" (de inventario) para poder comprar una antorcha. Sin ella no le serà posible explorar el castillo.

También debe tener en cuenta que debe comprar algo de comida. La comida es una posesión muy valiosa. Cuando comienza el programa tiene una fuerza limitada, que se va consumiendo gradualmente a medida que avanza el juego. Si llega a cero, usted muere y el juego termina. Las luchas contra los monstruos también disminuyen su fuerza, mientras que la comida se la vuelve a aumentar. Debe vigilar su medida de fuerza a lo largo del juego y siempre debe asegurarse que tiene comida a mano para recuperar sus energías cuando lo necesite.

Mapa del Entorno

Como le dije anteriormente, el entorno de la aventura debe ser coherente. Esto es, el explorador debe poder dibujar un mapa completo a medida que avanza a través del entorno. Si dibuja una puerta conectando el estudio con la biblioteca en un plano del suelo, ya que ha descubierto que a través de la puerta del estudio va a parar a la biblioteca, se supone que volviendo hacia atrás desde la biblioteca debe terminar de nuevo en el estudio. El jugador debe ser capaz de construir un plano completo de este modo, revisandolo de vez en cuando mientras recorre la casa, castillo, bosque laberinto subterràneo o cualquier sitio donde la aventura tenga lugar.

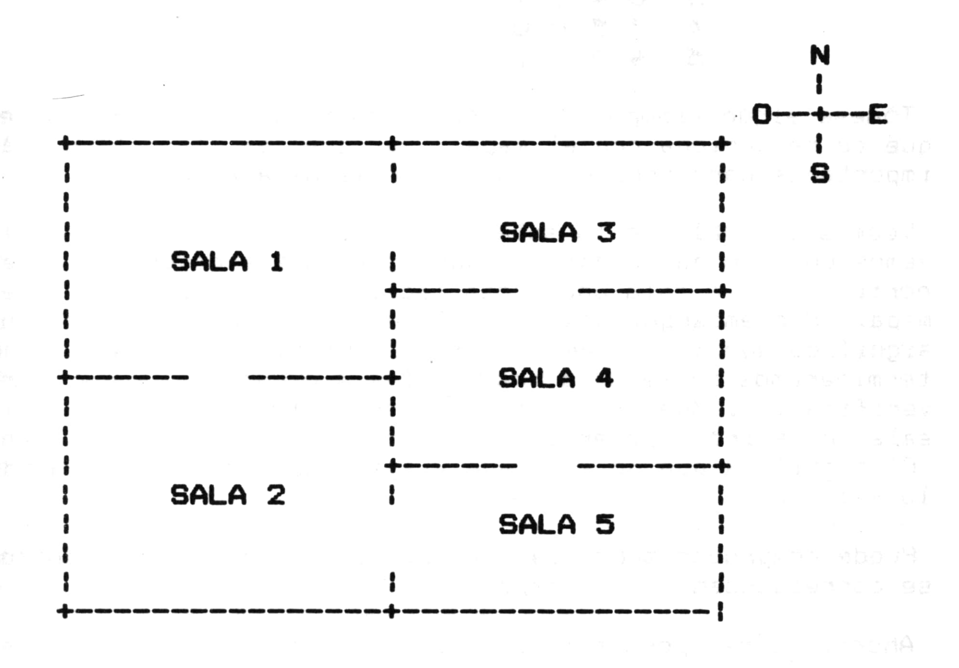
Por lo tanto, el primer paso al construir un programa de aventuras es preparar un entorno que pueda ser planificado y representado de manera que el ordenador lo pueda almacenar.

Verà con satisfacción que estas dos condiciones son fàciles de cumplir.

Eche una ojeada al siguiente diseño de una aventura con cinco salas, es muy simple pero nos va a servir para demostrarle còmo funciona en el ordenador.

La clave para que el ordenador pueda entender y manipular un entorno como èste, es crear una matrîz en la que cada

elemento represente una sala. Las aberturas entre las salas son las puertas.



Si estuvieramos en la sala 1, podrîamos movernos hacia el este a la sala 3, o hacia el sur a la sala 2. En la sala 4 nos podriamos mover al norte a la sala 3 y al sur a la sala 5, y asî sucesivamente. Imagînese que hemos preparado una matriz que ha sido dimensionada como DIM A(5,4). La primera dimensiòn es la sala y la segunda son las cuatro posibles direcciones desde esa sala (esto es, norte, sur, este y oeste).

Armados con el mapa de las cinco salas, podemos construir una *tabla del viaje*, que puede ser introducida en la matriz, para permitirnos movernos de un punto a otro dentro del entorno. He aquî la tabla de viaje de nuestro ejemplo:

SALA	N	S	E	O
1	0	2	3	0
2	1	0	5	0
3	0	4	0	1
4	3	5	0	0
5	4	0	0	2

Tòmese algùn tiempo estudiando esta tabla y la forma en que se relacciona con el mapa, ya que es la clave màs importante para construir un programa de aventuras.

Veamos la tabla de la sala 1. Bajo la columna 'N' (norte), vemos un cero que significa que no se puede mover hacia el norte desde la sala uno (cosa fàcil de verificar mirando el mapa). Sin embargo, bajo la 'S' vemos el nùmero dos que significa que si viajamos hacia el sur desde la sala uno terminaremos en la sala dos (vuelva a mirar el mapa para verificarlo). Muèvase hacia el este (columna 'E') desde la sala uno,e irà a parar a la sala tres. El 0 en la columna 'O' significa que no se puede viajar hacia el oeste desde la sala uno.

Puede comprobar sobre la tabla, que todos los nùmeros se corresponden con el mapa.

Ahora, para permitir al jugador moverse dentro del entorno, solamente necesitamos, (a) llenar cada elemento de la matriz con la informaciòn correspondiente de nuestra tabla de viaje; (b) decirle al jugador dònde se encuentra; y (c) permitirle que introduzca sus decisiones con la direcciòn hacia la que se quiere mover y verificarla contra nuestra matriz, entonces -si es posible- actualizarla para reflejar la nueva situaciòn del jugador. Es màs fàcil de hacer de lo que usted piensa.

Movimiento

Primero necesitamos escribir un pequeño programa para introducir la informaciòn dentro de la matriz. Dos simples bucles READ/DATA como los siguientes, lo haràn:

```
100 DIN a(5,4)
110 RESTORE
120 :
130 FOR b=1 TO 5
140  FOR c=1 TO 4
150   READ a(b,c)
160  END FOR c
170 END FOR b
180 :
190 DATA 0,2,3,0
200 DATA 1,0,5,0
210 DATA 0,4,0,1
220 DATA 3,5,0,0
230 DATA 4,0,0,2
```

Como puede ver, las sentencias DATA corresponden exactamente con los elementos de nuestra tabla de viaje.

Situaciòn del Jugador

Si decidimos que la sala que ocupa en cada momento el jugador viene representada en la variable RO, podemos decirle al jugador dònde està de la siguiente forma, asì como indicarle las salidas que existen:

```
100 PRINT "Estàs en la sala nùmero ";
ro
110 IF a(ro,1)<>0 THEN PRINT "Hay una
 puerta al norte"
120 IF a(ro,2)<>0 THEN PRINT "Hay una
 salida al sur"
130 IF a(ro,3)<>0 THEN PRINT "Puedes
salir por el este"
140 IF a(ro,4)<>0 THEN PRINT "hay un
camino al oeste"
```

Los datos que introduzca el jugador pueden ser una letra simple ('N' para el norte) y el programa puede recoger este e dato y comprobarlo para ver si existe una salida en esa direcciòn:

```
790  IF (mo$="n" OR mo$="N") AND a(ro
,1)=0 THEN PRINT " No existe esa sali
da":GO TO 650
800  IF (mo$="s" OR mo$="S") AND a(ro
,2)=0 THEN PRINT " No hay salida al s
ur":GO TO 650
810  IF (mo$="e" OR mo$="E") AND a(ro
,3)=0 THEN PRINT " No puedes ir en es
a dirección":GO TO 650
820  IF (mo$="o" OR mo$="O") AND a(ro
,4)=0 THEN PRINT " No puedes atravesa
r las piedras":GO TO 650
830  IF (mo$="a" OR mo$="A") AND a(ro
,5)=0 THEN PRINT "No puedes subir des
de aquí":GO TO 650
840  IF (mo$="b" OR mo$="B") AND a(ro
,6)=0 THEN PRINT "No puedes descender
 desde aquí":GO TO 650
```

Consistencia y Realidad

Aunque las salas solo existen en el papel y en los elementos de una matriz, el hecho de que vayan a ser como 'salas reales' ràpidamente nos permite percibirlas como si fueran reales y sòlidas de una forma misteriosa. Añada la descripciòn de cada sala -ESTA EN UNA HUMILDE CHOZA EN LA PARTE TRASERA DE UNA MANSION, HAY UNA PILA DE PAJA EN UNA ESQUINA ALEJADA Y UNA HORCA Y UN HACHA JUNTO A LA VENTANA. SOBRE LA MESA SE VE UN GRAN TROZO DE CARNE Y A SU LADO UNA NOTA. LAS PUERTAS ESTAN SITUADAS HACIA EL NORTE Y EL OESTE- Y encontrarà que el entorno toma unas dimensiones màs sòlidas en su imaginaciòn.

A medida que el mapa se hace màs complejo, como en *Los Lobos Fantasmas y el Vagabundo*, y las descripciones le ayudan a clarificar las imagenes mentales de las salas, comprobarà que tiene entre sus manos una simulaciòn de la realidad de un inmenso poder.

Puede intentar escribir un programa simple antes de

seguir adelante, que le permita moverse alrededor del entorno de cinco salas, como el que acabamos de ver.

Comandos

Aquí tenemos los comandos que deve usar para conducirse en esta aventura:

L - luchar
H - huir de un monstruo (no siempre podrà)
Q - avandonar la aventura (cobarde!)
N - moverse hacia el norte
S - moverse hacia el sur
E - moverse hacia el este
O - moverse hacia el oeste
A - moverse hacia arriba (el castillo tiene tres niveles)
B - moverse hacia abajo
I - inventario (para comprar una antorcha y otras cosas)
C - comer
T - tomar un tesoro
M - invocar el amuleto màgico para moverse a otro lugar (solamente se puede usar una vez, asi que consèrvelo hasta que lo necesite realmente)

Su puntuaciòn final estarà relaccionada con varios factores, incluido el tiempo de supervivencia, cuanta fuerza le queda, los tesoros encontrados y si todavìa lleva el amuleto màgico.

Si se siente suficientemente valiente, introduzca y ejecute la aventura de los *Lobos Fantasmas*.

```
10 REMeark Los lobos Fantasmas
20  RESTORE :CLEAR
30  empezar
40 REPeat bucle
50   rutina_principal
60    IF ro=11 THEN EXIT bucle
70 END REPeat bucle
80 REMark --- Fin del juego ---
90 PRINT \"  Lo has conseguido!!"
```

```
100 espera
110 PRINT \" Esa era la salida del ca
stillo"
120  espera
130 PRINT \\" Has tenido èxito, ";nom
bre$;"!"
140 PRINT\" saliendo del castillo"
150  espera
160 PRINT \"        Bien hecho!"
170  espera
180 PRINT \"  Tu puntuaciòn es ";
190 PRINT 39*cuenta + 51*fuerza + 21*
riqueza + comida + 309*mk + 45*traje
+100*amuleto + 17*espada + 7*hacha
200 STOP
210 REMark ------------------
220 DEFine PROCedure rutina_principal
230  fuerza=fuerza-5
240  IF fuerza<25 AND fuerza>=1 THEN
250   PRINT "   Peligro, ";nombre$;",
 tu fuerza"
260   PRINT "   te està abandonando!
"
270  END IF
280  IF fuerza<1 THEN
290   PRINT " Tu fuerza te ha abando
nado..."\"    y ahora estàs muerto...
"
300   GO TO 170
310  END IF
320  cuenta=cuenta+1
330  PRINT \\" ";nombre$;", tu fuerza
 es ";fuerza
340  IF riqueza>0 THEN PRINT "  Tiene
s ";riqueza;"$"
350  IF comida>0 THEN PRINT " Tu saco
 de provisiones contiene"\"          "
;comida;" unidades de comida"
360  IF traje=1 THEN PRINT " Llevas p
uesta la armadura"
370  IF hacha<>0 OR espada<>0 OR amul
eto<>0 THEN
```

```
380   PRINT " Llevas ";
390   IF hacha=1 THEN PRINT "un hacha
 ";
400   IF espada=1 THEN PRINT "una esp
ada ";
410   IF espada+hacha>0 AND amuleto=1
 THEN PRINT " y ";
420   IF amuleto=1 THEN PRINT !"el"!"
amuleto"!"màgico"
430  END IF
440 PRINT
450  IF luz=0 THEN
460   PRINT "Està demasiado oscuro pa
ra poder ver"
470  ELSE
480   descripcion_sala
490  END IF
500  k=a(ro,7)
510   IF k>9 THEN INK 2:PAPER 7:PRINT
 \" Hay un tesoro que vale ";k;"$":IN
K 6:PAPER 0:END IF
520  IF k<0 THEN
530   BEEP 1000,1:PAUSE 10:BEEP 1000,
2:PAUSE 10:BEEP 1000,3
540   INK 0:PAPER 5
550   PRINT \\\"Peligro!!! Hay un mon
struo aquì...":espera
560   IF k=-1 THEN m$="Feroz Lobo Fan
tasma":ff=5
570   IF k=-2 THEN m$="Fanàtico Estra
ngulador":ff=10
580   IF k=-3 THEN m$="Malèvolo Zombi
e":ff=15
590   IF k=-4 THEN m$="Devastador Dra
gòn Helado":ff=20
600   PRINT \"  Este es un ";m$
610   PRINT \"  El nivel de peligro e
s ";ff;"!!"
620   PRINT "
           "
630    INK 6:PAPER 0
640   END IF
```

```
650  espera
660  FLASH 1
670  PRINT \\" Què vas a hacer? "
680  FLASH 0
690  REPEAT mover
700   mo$=INKEY$
710   IF k<0 AND (mo$<>"l" AND mo$<>"
L" AND mo$<>"h" AND mo$<>"H") THEN GO
 TO 700
720   IF mo$<>"" THEN EXIT mover
730  END REPeat mover
740  INK 1
750  PRINT \"-------------------
--------------"
760  INK 6
770  PRINT
780  IF (mo$="q" OR mo$="Q") THEN GO
TO 180
790  IF (mo$="n" OR mo$="N") AND a(ro
,1)=0 THEN PRINT " No hay salida por
ese camino":GO TO 650
800  IF (mo$="s" OR mo$="S") AND a(ro
,2)=0 THEN PRINT " No hay salida por
el sur":GO TO 650
810  IF (mo$="e" OR mo$="E") AND a(ro
,3)=0 THEN PRINT " No se puede ir en
esa direcciòn": GO TO 650
820  IF (mo$="o" OR mo$="O") AND a(ro
,4)=0 THEN PRINT " No puedes atravesa
r las piedras":GO TO 650
830  IF (mo$="a" OR mo$="A") AND a(ro
,5)=0 THEN PRINT " No puedes subir de
sde aquì":GO TO 650
840  IF (mo$="b" OR mo$="B") AND a(ro
,6)=0 THEN PRINT " No puedes descende
r desde aquì":GO TO 650
850  IF (mo$="h" OR mo$="H") AND RND>
.7 THEN
860   PRINT " No, tienes que quedarte
 y luchar"
870   mo$="l"
```

```
880   espera
890  END IF
900  IF mo$="h" OR mo$"H" THEN
910   PRINT " Hacia donde quieres cor
rer?"
920   espera
930   k=0
940   mo$=""
950   GO TO 680
960  END IF
970  IF (mo$="l" OR mo$="L") AND a(ro
,7)>-1 THEN
980   PRINT " Aquí no hay nadie con q
uien luchar!"
990  GO TO 650
1000 END IF
1010  IF mo&="i" OR mo$="I" THEN inve
ntario
1020  IF (mo$="c" OR mo$="C") AND com
ida=0 THEN
1030   PRINT " No tienes comida!"
1040   GOTO 650
1050  END IF
1060  IF mo$="t" OR mo$="T" THEN teso
ro
1070  IF mo$="l" OR mo$="L" THEN luch
ar
1080  IF mo$="m" OR mo$="M" THEN
1090   REPeat elegir
1100    ro=RND(1 TO 19)
1110     IF ro<>6 AND ro<>11 THEN EXIT
 elegir
1120   END REPeat elegir
1130   amuleto=0:REMark El amuleto
                          solo se usa
                          una vez
1140  END IF
1150  IF mo$="c" OR mo$="C" THEN come
r
1160  IF mo$="n" OR mo$="N" THEN ro=a
(ro,1)
1170  IF mo$="s" OR mo$="S" THEN ro=a
```

```
(ro,2)
1180  IF mo$="e" OR mo$="E" THEN ro=a
(ro,3)
1190  IF mo$="o" OR mo$="O" THEN ro=a
(ro,4)
1200  IF mo$="a" OR mo$="A" THEN ro=a
(ro,5)
1210  IF mo$="b" OR mo$="B" THEN ro=a
(ro,6)
1220 END DEFine rutina_principal
1230 REMark -----------------
1240 DEFine PROCedure luchar
1250  FOR j=1 TO 30
1260   BEEP 100,j:BEEP 100,200-j
1270  END FOR j
1280  IF INKEY$<>"" THEN GO TO 1280
1290  FLASH 1:PRINT \"  Pulse una tec
la para luchar...":FLASH 0
1300  REPeat teclea
1310   IF INKEY$<>"" THEN EXIT teclea
1320  BEEP 100,5:END REPeat teclea
1330  FOR j=1 TO 6
1340   INK RND(1 TO 7)
1350   PRINT \" * - * - * - * - * - *
 - * - *"
1360  END FOR j
1370  INK 6
1380  IF traje=1 THEN
1390   PRINT " Tu armadura aumenta tu
s              probabilidades de èxito"
1400   ff=3*(INT(ff/4))
1410   espera
1420  END IF
1430  IF hacha=0 AND espada=0 THEN
1440   PRINT " No tienes armas y debe
s luchar con  las manos desnudas"
1450   ff=INT(ff+ff/5)
1460   espera
1470  END IF
1480  IF hacha<>0 OR espada<>0 THEN
1490   IF hacha=1 AND espada=0 THEN
1500    PRINT " Solo tienes un hacha
```

```
para luchar"
1510     ff=4*INT(ff/5)
1520    END IF
1530    IF hacha=0 AND espada=1 THEN
1540     PRINT " debes luchar con
                    tu espada"
1550     ff=3*INT(ff/4)
1560    END IF
1570    IF hacha=1 AND espada=1 THEN
1580     PRINT " Que arma? 1 - Hacha
  2 - Espada"
1590     REPeat elegir
1600      arma$=INKEY$
1610      IF arma$="1" OR arma$="2" TH
EN EXIT elegir
1620     END REPeat elegir
1630    z=arma$
1640    IF z=1 THEN ff=4*INT(ff/5)
1650    IF z=2 THEN ff=4*INT(ff/4)
1660   END IF
1670   END IF
1680   REMark -- La Batalla --
1690   PRINT \
1700  REPeat follon
1710   PRINT:FLASH RND (0 TO 1)
1720    INK RND(3 TO 7)
1730    BEEP 1000,RND(10 TO 212)
1740    BEEP 1000,RND(10 TO 212)
1750   IF RND>.5 THEN
1760    PRINT " El ";M$;" ataca"
1770     ELSE
1780    PRINT "Tu atacas al ";m$
1790   END IF
1800    espera
1810   IF RND>.5 THEN
1820    PRINT " Estås ganandole!"
1830    ff=INT(5*ff/6)
1840   END IF
1850    espera
1860   IF RND>.5 THEN
1870    PRINT " El monstruo te ha gana
do!"
```

```
1880    fuerza=fuerza-5
1890   END IF
1900   IF RND<.35 THEN EXIT follon
1910 END REPeat follon
1920    INK 6
1930  IF RND>ff/10 THEN
1940   PRINT \" Estàs matando al
                 ";m$;"!!"
1950   mk=mk+1
1960   fuerza=fuerza+INT(fuerza/2)
1970    ELSE
1980   PRINT \" El ";m$!" te ha "!"de
rrotado..."
1990   fuerza=INT(fuerza/2)
2000  END IF
2010  a(ro,7)=0:espera:espera
2020  FLASH 0
2030 END DEFine lucha
2040 REMark ---------------
2050 DEFine PROCedure decripcion-sala
2060       INK 2
2070 PRINT \" ^V^V^V^V^V^V^V^V^V^V^V^
V^V^V^V^V^
              "
2080      INK 6
1090 SELECT ON RO
2100 ON RO=1
2110 PRINT " Estàs en el vestìbulo"
2120 PRINT " Hay una puerta al sur, y
"
2130 PRINT " a travès de las ventanas
 del norte "
2140 PRINT " puedes ver un jardìn sec
reto."
2150 ON ro=2
2160 PRINT " Esta es la Sala de Audie
ncias."
2170 PRINT " Hay una ventana al oeste
.  "
2180 PRINT " Hacia la derecha puedes
ver"
2190 PRINT " la entrada al castillo."
```

```
2200 PRINT " Hay puertas para hacia e
l norte                   este y sur"
2210 ON ro=3
2220 PRINT " Estàs en el Gran Salòn,
una sala"
2230 PRINT " en forma de L. Hay puert
as al este"
2240 PRINT " y al norte. Hay"
2250 PRINT " una puerta al oeste en
                         la alcoba"
2260 ON ro=4
2270 PRINT " Esta es la sala privada
de "
2280 PRINT " Audiencias del Monarca.
hay una sola             salida al sur"
2290 ON ro=5
2300 PRINT " Este pequeño recibidor t
iene una"
2310 PRINT " puerta al norte, y otra
al oeste,"
2320 PRINT " y una escalera de caraco
l"
2330 PRINT " pasa a travès de la sala
. Puedes"
2340 PRINT " ver un lago ornamental a
 travès de"
2350 PRINT "       las ventanas del su
r"
2360 ON ro=6
2370 PRINT " Estàs en la entrada del
"
2380 PRINT " castillo de piedra prohi
bido."
2390 ON ro=7
2400 PRINT " Esta es la cocina del ca
stillo."
2410 PRINT " Por las ventanas del mur
o norte"
2420 PRINT " puedes ver un jardìn sec
reto."
2430 PRINT " Una puerta sale de la co
cina hacia                   el sur"
```

```
2440 ON ro=8
2450 PRINT " Estàs en el almacèn, rod
eado de"
2460 PRINT "especias, verduras, grand
es sacos"
2470 PRINT "de harina y otras provisi
ones."
2480 PRINT "Hay una puerta en el nort
e y            otra en el sur"
2490 ON ro=9
2500 PRINT " Has entrado en un ascens
or..."
2510  espera
2520 PRINT "que desciende lentamente.
.."
2530  espera:ro=10
2540 GO TO 3080
2550 ON ro=10
2560 PRINT " Estàs en el vestìbulo de
 atràs."
2570 PRINT " Hay unas ventanas que da
n al sur"
2580 PRINT " desde la que puedes ver
el"
2590 PRINT " lago ornamental. Hay una
 salida"
2600 PRINT al este y otra al norte."
2610 ON ro=12
2620 PRINT " Estàs en un hùmedo t osc
uro calabozo."
2630 PRINT " Solo hay una salida, un
pequeño"
2640 PRINT " agujero en el muro hacia
 el oeste."
2650 ON ro=13
2660 PRINT " Te encuentras en la sala
 de la"
2670 PRINT " guardia de la prisiòn, e
n el sòtano"
2680 PRINT " del castillo. La escaler
a de caracol"
```

```
2690 PRINT " termina en esta sala. Ha
y otra"
2700 PRINT " salida, un pequeño aguje
ro en el                  muro este."
2710 ON ro=14
2720 PRINT " Estàs en el dormitorio p
rincipal en"
2730 PRINT " el piso superior del cas
tillo..."
2740 PRINT " Mirando desde la ventana
 hacia el"
2750 PRINT " oeste, puedes ver la ent
rada del"
2760 PRINT " castillo, mientras que e
l jardìn"
2770 PRINT " secreto es visible desde
 la ventana"
2780 PRINT " norte. Hay puertas al  "
2790 PRINT "         este y al sur.
2800 ON ro=15
2810 PRINT " Este es el vestìbulo sup
erior."
2820 PRINT " Hay una puerta al norte,
 y"
2830 PRINT " puede ver tambièn una es
calera."
2840 PRINT " A travès de la ventana s
ur"
2850 PRINT " le llega el reflejo del
lago."
2860 ON ro=16
2870 PRINT " Esta sala fuè usada como
 sala del"
2880 PRINT " tesoro del castillo hace
 muchos años"
2890 PRINT " No tiene ventanas, solo
salidas"
2900 PRINT " al este y al norte"
2910 ON ro=17
2920 PRINT " Ooh!! Estàs en el dormit
orio de las"
2930 PRINT " doncellas. Hay una salid
```

```
a al"
2940 PRINT " oeste y una puerta al su
r"
2950 ON ro=18
2960 PRINT " Esta pequeña habitaciòn
del piso"
2970 PRINT " superior Es el Vestidor.
 Hay una"
2980 PRINT " ventana al norte, con un
a vista del"
2990 PRINT " jardìn bajo ella."
3000 PRINT "   Una puerta t Oe lleva
al sur."
3010 ON ro=19
3020 PRINT " Esta es una pequeña sala
 fuera del"
3030 PRINT " ascensor del castillo, a
 la que se"
3040 PRINT " accede por una puerta al
 norte. Otra"
3050 PRINT " puerta queda al oeste. P
uedes"
3060 PRINT " ver el lago a travès de"
3070 PRINT " las ventanas del sur."
3080 END SELect
3090 END DEFine descripcion_sala
3100 REMark ----------------
3110 DEFine PROCedure tesoro
3120  FOR j=1 TO 60
3130   BEEP 101,j
3140  END FOR j
3150  IF a(ro,7)<10 THEN
3160   PRINT "No hay tesoro que cojer
"
3170   espera
3180  END IF
3190  IF luz=0 THEN
3200   PRINT " No puedes ver donde es
tà"
3210   espera
3220 END IF
3230  IF luz=1 AND a(ro,7)>9 THEN
```

```
3240    riqueza=riqueza+a(ro,7)
3250    a(ro,7)=0
3260    PRINT \" Has cogido el tesoro!
"
3270    espera
3280   END IF
3290 END DEFine tesoro
3300 REMark ----------------
3310 DEFine PROCedure comer
3320  FOR j=1 TO 20
3330    BEEP 500,200+j:BEEP 500,200-j
3340   END FOR j
3350   CLS
3360   IF comida>1 THEN
3370    PRINT " Tienes ";comida;" unid
ades de comida"
3380    PRINT " Cuanto quieres comer?"
3390    REPeat como-comida
3400     PRINT "        ";
3410     INPUT z
3420     IF z<=comida THEN EXIT comer_
comida
3430    END REPeat como_comida
3440    comida=INT(comida-z)
3450    fuerza=INT(fuerza+5*z)
3460    espera
3470   END IF
3480 END DEFine comer
3490 REMark ---------------
3500 DEFine PROCedure empezar
3510   BORDER 0,0:PAPER 0:INK 6
3520   CLS:CLS #0
3530   fuerza=100
3540   riqueza=75
3550   comida=0
3560   cuenta=0
3570   mk=0:REMark Número de monstruos
                             muertos
3580 REMark ---------------
3590 REMark Preparar castillo
3600 DIMa(19,7)
3610 FOR b=1 TO 19
```

```
3620  FOR c=1 TO 7
3630   READ a(b,c)
3640  END FOR c
3650  BEEP 1000,5*b
3660 END FOR b
3670 INPUT \\" Cual es tu nombre, exp
lorador?

                         ";nombre$
3680 CLS
3690 ro=6:REMark Posiciòn de partida
                  (ro=nùmero de sala)
3700 espada=0
3710 amuleto=0
3720 traje=0
3730 lus=0
3740 hacha=0
3750 REMark ----------------
3760 REMark Repartir el tesoro
3770 FOR j=1 TO 4
3780  REPeat repartir
3790   m=RND(1 TO 19)
3800   IF m<>6 AND m<>11 AND a(m,7)=0
 THEN EXIT repartir
3810  END REPeat repartir
3820  a(m,7)=RND(10 TO 110)
3830 END FOR j
3840 REMark ----------------
3850 REMark Repartir monstruos
3860 FOR j=1 TO 4
3870  REPeat repartir
3880   m=RND(1 TO 18)
3890   IF m<>6 AND m<>11 AND a(m,7)=0
 THEN EXIT repartir
3900  END REPeat repartir
3910  a(m,7)=-j
3920 END FOR j
3930  a(4,7)=RND(100 TO 199)
3940  a(16,7)=RND(100 TO 199)
3950 END DEFine empezar
3960 REMark ---------------
3970 DEFine PROCedure inventario
```

```
3980 trampa=0
3990 PRINT " Provisiones e Inventario
"
4000 REPeat provisiones
4010 espera
4020  IF riqueza=0 THEN EXIT provisio
nes
4030 IF riqueza=.1 THEN
4040  PRINT \\"     Tienes ";riqueza;
"$"
4050  PRINT " Puedes comprar 1 - Anto
rcha (15$)"
4060  PRINT "                  2 - Hach
a (10$)"
4070  PRINT "                  3 - Espa
da (20$)"
4080  PRINT "                  4 - Comi
da (2$ unidad)"
4090  PRINT "                  5 - Amul
eto (30$)"
4100  PRINT "                  6 - Arma
dura (50$)
4110  PRINT "                  0 - Cont
inuar juego"
4120  FLASH 1
4130 PRINT \"    Elija un nùmero"
4140  FLASH 0
4150 IF INKEY$<>"" THEN GO TO 4150
4160 REPeat num
4170  z$=INKEY$
4180  IF z$>="0" AND z$<="6" THEN EXI
T num
4190 END REPeat num
4200 z=z$
4210 IF z=0 THEN EXIT provisiones
4220 IF riqueza<0 AND z<>0 THEN tramp
a=1:EXIT provisiones
4230 SELect ON z
4240 ON z=1
4250  luz=1
4260  riqueza=riqueza-15
4270 ON z=2
```

```
4280  hacha=1
4290  riqueza=riqueza-10
4300 ON z=3
4310  espada=1
4320  riqueza=riqueza-20
4330 ON z=5
4340  amuleto=1
4350  riqueza=riqueza-30
4360 ON z=6
4370  traje=1
4380  riqueza=riqueza-50
4390 ON z=4
4400  PRINT " Cuantas unidades de com
ida?"
4410   REPeat cuantas
4420    PRINT "     ";
4430    INPUT muchas
4440    IF 2*muchas<=riqueza THEN EXI
T cuantas
4450   END REPeat cuantas
4460   muchas=INT(muchas)
4470   comida=comida+muchas
4480   riqueza=riqueza-2*muchas
4490 END SELect
4500 END IF
4510 END REPeat provisiones
4520 IF trampa=1 THEN
4530  PRINT " Has tratado de engañarm
e!!"
4540  riqueza=0:traje=0:hacha=0:espad
a=0
4550  amuleto=0:comida=INT(comida/4)
4560  espera
4570 END IF
4580 END DEFine inventario
4590 REMark ---------------
4600 DEFine PROCedure espera
4610  IF INKEY$<>"" THEN GO TO 4610
4620  BEEP 14000,RND(1 TO 50)
4630  PAUSE 23
4640  BEEP 32000,RND(60 TO 112),255,4
4650  PAUSE 23
```

```
4660  BEEP 32000,RND(220 TO 250),255,
4,5,7,0,RND(0 TO 15)
4670  FOR paciencia=1 TO 500
4680  END FOR paciencia
4690 END DEFine espera
4700 REMark ---------------
4710 DATA 0,2,0,0,0,0,0 REMark sala 1
4720 DATA 1,3,3,0,0,0,0 REMark  2
4730 DATA 2,0,5,2,0,0,0 REMark  3
4740 DATA 0,5,0,0,0,0,0 REMark  4
4750 DATA 4,0,0,3,15,13,0 REMark  5
4760 DATA 0,0,1,0,0,0,0 REMark  6
4770 DATA 0,8,0,0,0,0,0 REMark  7
4780 DATA 7,10,0,0,0,0,0 REMark  8
4790 DATA 0,19,0,8,0,8,0 REMark  9
4800 DATA 8,0,11,0,0,0,0 REMark  10
4810 DATA 0,0,10,0,0,0,0 REMark  11
4820 DATA 0,0,0,13,0,0,0 REMark  12
4830 DATA 0,0,12,0,5,0,0 REMark  13
4840 DATA 0,15,17,0,0,0,0 REMark  14
4850 DATA 14,0,0,0,0,5,0 REMark  15
4860 DATA 17,0,19,0,0,0,0 REMark  16
4870 DATA 18,16,0,14,0,0,0 REMark  17
4880 DATA 0,17,0,0,0,0,0 REMark  18
4890 DATA 9,0,16,0,0,0,0 REMark  19
```

Una vez que estè suficientemente familiarizado con la forma en que està construida esta aventura, puede usar el programa que tiene en el microdrive para crear otras aventuras por sì mismo, usando el programa como una rutina principal dentro de su propio mapa, descripciòn de salas y nombres de monstruos y criaturas.

Soporte e Ideas

Existe una gran cantidad de literatura sobre juegos de Aventuras. La selecciòn que le damos a continuaciòn es el producto de mi interès en este campo y no es una selecciòn de las mejores publicaciones. Sin embargo, la lista se ha hecho con los trabajos que yo he encontrado màs interesan-

tes. Debe haber probablemente cientos de igual mèrito, pero al menos esta lista le puede dar un punto de partida:

THROUGH DUNGEONS DEEP: A Fantasy Gamer's Handbook - Robert Plamondon (Reston Publishing Company, Inc., Reston, Virginia, 1982).

What is Dungeons and Dragons® - Jhon Butterfield, Philip Parker y David Honigman (Penguin Books Ltd., Harmondsworth, Middlesex, England, 1982). * Dungeons and Dragons es un juego de aventuras original de TSR Hobbies Inc. TM El tèrmino es una marca registrada.

Dicing with Dragons, an Introduction to Role-Playing Gamees - Ian Livingstone (Routledge & Kegan Paul, London,Melbourne and Hengley, 1982).

Fantasy Role Playing Games - J Eric Holmes (Hippocrane Books, Inc., New York, 1981).

Hay tambièn un gran nùmero de ayudas para construir mapas del entorno donde desarrollar la aventura. Entre ellos està:

DUNGEONS & DRAGONS® las cajas de juegos son una buena forma de aprender algunas de las posibilidades de los juegos de Aventuras. Puede empezar (y continuar si le interesa) con el *BASIC SET WITH INTRODUCTORY MODULE*. Este juego contiene dos libros y seis dados con varios nùmeros en cada cara. Los libros son las *REGLAS BASICAS* que le dan los conceptos y las normas del juego y le explican los caracteres y las personalidades de los personajes, còmo luchar y muchos otros aspectos de las reglas del juego.

Ademàs de los libros y los dados, el juego contiene un mapa de campaña, *THE KEEP ON THE BORDERLANDS*. Contiene tambièn gran cantidad de informaciòn, incluido una serie de mapas, informaciòn de las salas y màs detalles sobre la resoluciòn de las luchas. Creo que esta caja es probablemente la mejor fuente de ideas que puede tener en sus manos. Es tambièn una buena forma de conseguir comprender còmo se han desarrollado y como se controlan las

reglas de este tipo de juegos.

Estas fuentes de ideas se han sugerido, por supuesto, solamente para su propio uso, para producir juegos para su entretenimiento. No le està permitido incorporar material registrado para juegos de venta al pùblico.

Las siguientes casas distribuyen material para aventuras y juegos que le pueden dar màs ideas:

Avalon Hill Games, 650 High Street, North Finchley, London N12 ONL.
Citadel Miniatures, 10 Victoria Street, Newark, Nottinghamshire.
Games of Liverpool, 50-54 Manchester Street, Liverpool, L1 6ER.
Flying Buffalo, PO Box 100, Bath Street, Walsall, West Midlands.
Games Workshop =td., 1 Dalling Road, London, W6; 27-29 Sunbeam Road, London NW 10
Simpubs Ltd., Oakfield House, 60 Oakfield Road, Altrincham, Cheshire WA15 8EW.
TSR Hobbies (UK) Ltd., The Mill, Rathmore Road, Cambridge, CB1 4AD.

Apêndices

Apêndice Uno -
Demostración de Multi-tarea

Este programa, escrito por Derek Wilson, que se publicó por primera vez en la revista Quanta genera, sobre la pantalla, una imagen del reloj de tiempo real del QL, funcionando como una tarea independiente. Ejecute primero el siguiente programa con un cartucho, tratado con el comando FORMAT, en el microdrive 2. Esto genera un fichero en codigo de màquina llamado "reloj".

```
90 REMark   Multi-tarea - Reloj
91 REMark De QUANTA - Vol  1 Issue 8
92:
100 c=RESPR(100)
105 RESTORE
110 FOR i=0 TO 68 STEP 2
120  READ x:POKE_W i+c,x
130 END FOR i
140 SEXEC mdv2_reloj,c,100,256
1000 DATA 29439,29697,28683,20033,174
02
1010 DATA 48,13944,200,20115,12040
1020 DATA 28691,20033,17402,74,-27698
1030 DATA 13944,236,20115,8279,-11314
1040 DATA 13944,208,20115,16961,16962
1050 DATA 30463,28688,20035,24794
1060 DATA 0,7,240,10,272,200
```

Una vez en su lugar, teclèe:

```
EXEC mdv2_reloj
```

Esto producirà un reloj digital continuo en la esquina inferior derecha de la pantalla.

La lînea 1060 gobierna la posiciòn y el color de la ventana del reloj. Cambie el 200 por 272 si quiere que hacerlo funcionar en MODE 4. Los DATA de èsta lînea corresponden a lo siguiente:

BORDER color - 1 octeto

BORDER ancho - 1 octeto

PAPER color - 1 octeto

INK color - 1 octeto

Los datos introducidos con POKE_W, son pares de bytes o palabras. Para crear un nueva palabra para PAPER e INK, debe calcular el valor del DATA con la siguiente fòrmula:

256*PAPER+INK

Por ejemplo, si queremos PAPER blanco e INK roja nos resultarà 256*7+2=1794.

WINDOW ancho - 1 palabra (2 octetos)

WINDOW alto - 1 palabra

Origen X - 1 palabra

Origen Y - 1 palabra

Apêndice Dos -
Funciones Matemâticas

El QL contiene algunas de las funciones trigonomêtricas estândar. La lista que viene a continuaciòn las describe brevemente. Tenga en cuenta que los àngulos para las funciones trigonomètricas deben ser expresados en radianes.

COS - nos da el coseno de un nùmero en el rango -60000 a +60000. Para usarlo debe poner detràs de la palabra COS el argumento entre parêntesis: COS(564).

SIN - nos da el seno del argumento y, como el COS, opera en el rango -60000 a +60000. El formato es SIN(argumento).

TAN - TAN(argumento) nos darà la tangente del argumento, que debe estar comprendido entre -30000 y +30000.

ATAN y ACOT - se usan para determinar el arco de tangente y de cotangente de un argumento. A diferencia de las funciones anteriores, no tiene lîmite el tamaño del argumento (aparte de los lîmites propios del QL. se usan como ATAN(argumento) y ACOT(argumento).

COT - produce la cotangente del argumento. Funciona con un rango de -30000 a +30000. El formato es el mismo que en los anteriores: COT(argumento).

EXP - calcula el resultado de elevar el nùmero 'e' a la potencia del argumento. Para asegurarse que no rebasa los lîmites aritmèticos del QL, el argumento debe estar entre -500 y +500. Su formato es EXP(argumento).

LN y LOG10 - se usan para obtener logaritmos de los argumentos. LN nos da el logaritmo natural y LOG10 el logaritmo en base 10. El argumento debe ser positivo pero no tiene lîmite superior (aparte de los del QL).

INT - devuelve el valor de la parte entera del argumento, por ejemplo, 29 es INT(29.8756) y -28 para INT(-28.789). El argumento debe estar entre -32767 y + 32767. Si quiere redondear un nùmero hacia el entero superior, añàdale 0.5

como en INT(número + 0.5).

ABS - nos devuelve el valor absoluto del argumento, ignorando el signo, por lo que devuelve siempre un número positivo.

SQRT - nos da la raiz cuadrada del argumento, que debe ser mayor que cero. Para usarlo ponga SQRT(argumento).

Apêndice Tres - Como empezô Todo

Todo empezô con la carta que nos anunciaba la revelaciôn del QL:



sinclair

INVITATION

Sinclair Research Limited London

[illegible]

I have great pleasure in inviting you to attend a press conference, given by Clive Sinclair and Nigel Searle on Thursday 12 January, to launch Sinclair Research's new micro-computer.

It will take place in the Ballroom at the Hotel Inter-Continental, Hamilton Place, (off Park Lane), W1, starting at 10.30 prompt and be followed at approximately 11.45 by a detailed technical demonstration and discussion for those who wish to remain.

You are also very welcome to join us beforehand for breakfast which will be available from 9.30.

On arrival, please use the Ballroom entrance.

I apologise for the short-notice and do hope that you, or your representative, will be able to attend. It would be most helpful if you would RSVP either by returning the enclosed reply-paid card or by 'phoning.

PLEASE NOTE: The information contained in this invitation is confidential and EMBARGOED UNTIL 11am JANUARY 12.

Apêndice Cuatro - Una introducción al 68008

68008 ESTRUCTURA INTERNA

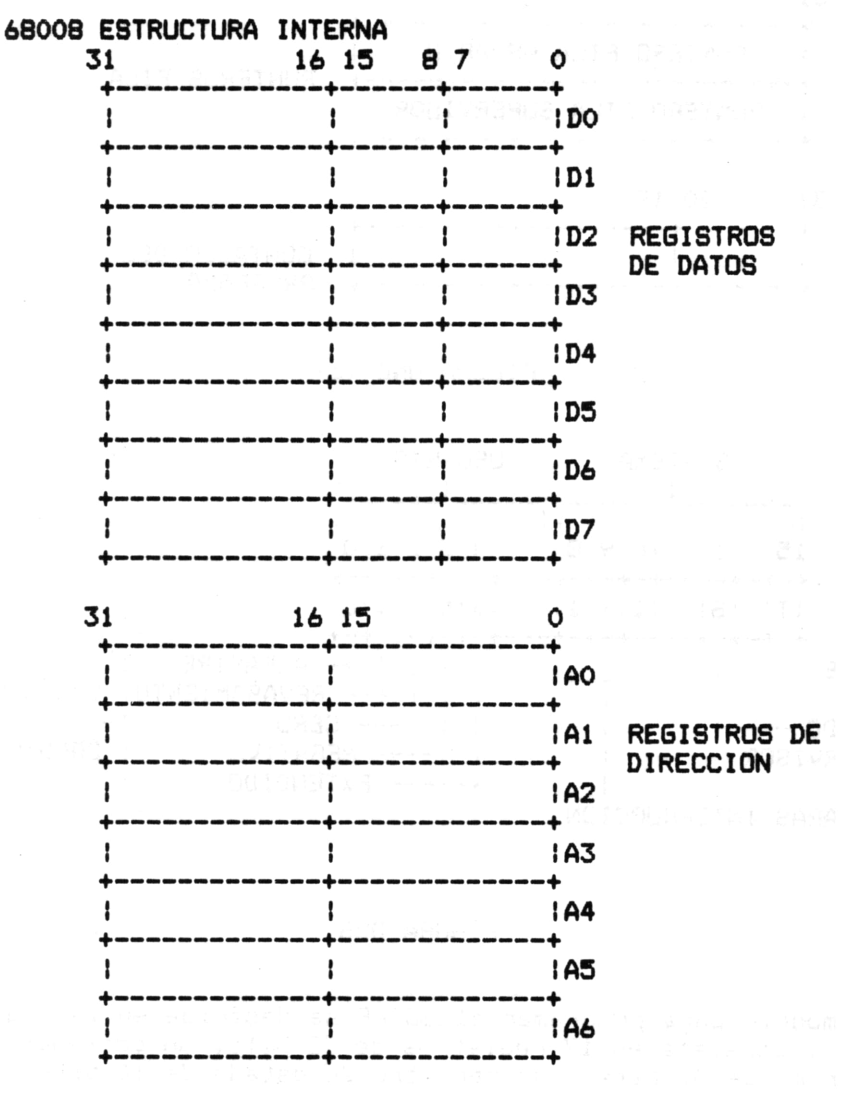


FIGURA UNO (a)

```
31                              0
 + - - - - - - - - - - - - - - -+
 !    PUNTERO PILA USUARIO      !
 +------------------------------+  PUNTEROS PILA
 !  PUNTERO PILA SUPERVISOR     !
 + - - - - - - - - - - - - - - -+

31       20 19                  0
 + - - - +----------------------+
 !       !                      !  CONTADOR DEL
 + - - - +----------------------+  PROGRAMA
```

FIGURA UNO (b)

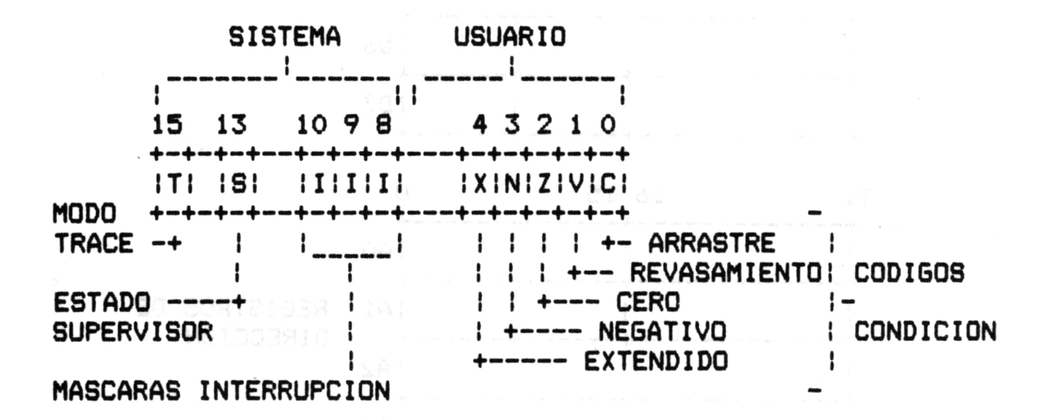


FIGURA DOS

El modelo para programar el 68008 se describe en la figura uno, y consiste en 17 registros de 32 bits, un contador de programa de 32 bits y un registro de estado de 16 bits.

Los primeros ocho registros (D0 a D7) se usan como registros de datos para operaciones de bit, octeto (8 bit), palabra (16 bit) y palabra larga (32 bit). El segundo grupo de siete registros (A0 a A6) y el puntero de la pila del sistema (A7) pueden ser usados como punteros de pila y registros base de direcciòn. El puntero de la pila del sistema se puede usar como puntero de la pila del supervisor (SSP) o como puntero de la pila del usuario (USP), dependiendo del estado del bit 13 (bit S) en el registro de estado.

El registro de estado (figura dos) puede considerarse como de dos octetos, uno de usuario y el otro de sistema. El octeto de usuario contiene cinco bits que definen los còdigos de condiciòn de arrastre (carry C), rebasamiento (overflow V), cero (zero Z), negativo (N) y extendido (X).

El octeto del sistema contiene cinco bits que definen la prioridad actual de las interrupciones (cuatro niveles de interrupciòn), mientras el procesador està en mode Trace, y/o en estado supervisor o usuario. En modo Trace, el ñrocesador irà al estado de proceso de excepciòn (como una interrupciòn de 'software') despuès de cada interrupciòn. Esto hace muy fàcil la escritura de herramientas para depuraciòn en modo paso-a-paso. El paso-a-paso por 'hardware' es tambièn muy fàcil de realizar.

El contador de programa es un registro de 32 bits, pero solamente 20 de ellos estàn disponibles en el caso del 68008, permitiendo un espacio de direcciones de 1Mb. La mayorìa de las instrucciones operan con octetos, palabras y palabras largas.

Hay 14 modos de direccionamiento disponibles:

REGISTRO DIRECTO
- **Registro Directo de Datos**
- **Registro directo de direcciòn**

DATO ABSOLUTO
- **Absoluto corto**
- **Absoluto Largo**

CONTADOR DE PROGRAMA

Relativo con Desplazamiento
Relativo con Indice y Desplazamiento
REGISTRO INDIRECTO
Registro Indirecto
Indirecto Post-incrementado
Indirecto Predecrementado
Indirecti con Desplazamiento
Indirecto Con Indice y Desplazamiento
DATO INMEDIATO
Inmediato
Inmediato Ràpido
IMPLICITO
Registro Implìcito

El juego de instrucciones està diseñado alrededor de un nùmero relativamente pequeño de nemotècnicos para hacer las cosas màs fàciles al programador. Sin embargo, el gran nùmero de modos de direccionamiento que se aplican a la mayorìa de las instrucciones (llamado juego de instrucciones ortogonal) nos da un total de 1000 instrucciones ùtiles. Estas incluyen la multiplicaciòn y divisiòn con o sin signo, operaciones aritmèticas ràpidas (con una cantidad de 8 bits como parte del còdigo de operaciòn) y aritmètica BCD. Por si esto no fuera suficiente, ciertos còdigos de operaciòn inician procesos de excepciòn, permitièndole definir sus propias instrucciones y simular su operaciòn en 'software'.

Si està interesado en leer algo màs, el 'hardware' del 68000/68008 està cubierto en dos publicaciones disponibles en los distribuidores de Motorola. Estos son el "16-bit Microprocessors Data Manual" (que incluye tambièn los 'chips' perifèricos de 8 bits) y los dos volùmenes de "MCU/MPU Applications Manual". Se pueden obtener gratuitamente las hojas de datos y notas de aplicaciòn incluidas en estas publicaciones, pero son màs ùtiles los libros para una referencia màs fàcil.

Le recomiendo tambièn el "Assembly Languaje Programming" por Kane, Hawkins y Levental, publicado por Osborne/Mc Graw-Hill para programar en còdigo màquina el 68000/68008.

Es un libro excelente, y no demasiado caro. Tambièn le recomiendo el "M68000 16/32-bit Microprocessor Programmer's Reference Manual", cuarta ediciòn, escrito por Motorola y publicado por Prentice-Hall.Este ùltimo se puede considerar como 'la Biblia' cuando no estê seguro sobre algo concerniente al juego de instrucciones del 68000/68008 Usàndolo, puede ser posible ensamblar a mano pequeños programas, debido a la estupenda documentaciòn que contiene acerca de los còdigos de operaciòn.

Las oficinas de ventas de Motorola le daràn una Tarjeta de Referencia para el Programador gratuitamente, y tengo entendido que tambièn proporcionan tarjetas con una microfotografìa del 68000, y un 'chip' (la pequeña pieza de silicona) montada en la tarjeta. Se supone que son dispositivos rechazados por el control de calidad.

Este apêndice ha sido escrito por Leon Heller, y està basado en el material que apareciò en 'NATGUG News', el periòdico de los usuarios del TRS-80 y el Genie, y en *Quanta*. Suyo es tambièn el libro *Understanding the 68008*.

Apêndice Cinco - Mensajes de Error

El SuperBASIC reporta los errores en formato estándar:

At line *número_línea texto_error*

Done número_línea es el número de la línea donde se ha detectado el error y texto_error es uno de los que se listan a continuación:

(1) Not complete
Una operación ha sido terminada prematuramente.

(2) Invalid job
El QDOS ha devuelto un error relacionado con las llamadas del sistema que controlan la multitarea o las operaciones de E/S.

(3) Out of memory
l QDOS y/o el SuperBASIC tienen insuficiente memoria libre.

(4) Out of range
Suele ser el resultado de un intento de escribir fuera de los límites de una ventana o un índice de matriz incorrecto.

(5) Buffer full
Una operación de E/S que debe llenar una memoria intermedia, la ha llenado antes de encontrar un fin de registro.

(6) Channel not open
Se ha intentado leer, escribir o cerrar un canal que no ha sido abierto previamente.
También puede ocurrir si falla un intento de abrir un canal.

(7) Not found
No se puede encontrar un Sistema de ficheros, dispositivo, medio o fichero.
El SuperBASIC no puede encontrar un identificador.

Puede ser el resultado de un nido de estructuras incorrecto.

(8) Already exists
El sistema de ficheros ha encontrado un fichero con el mismo nombre que el nuevo fichero que se va a abrir.

(9) In use
El sistema de ficheros ha encontrado que el fichero o dispositivo se està usando en exclusiva.

(10) End of file
Se ha detectado un fin de fichero durante la entrada de datos.

(11) Drive full
Un dispositivo Microdrive se ha llenado.

(12) Bad name
El sistema de ficheros ha reconocido un nombre pero hay un error de sintaxis o el valor de un paràmetro erròneo.
En SuperBASIC significa que se ha usado un nombre fuera del contexto. Por ejemplo, se ha usado una variable como un procedimiento.

(13) Xmit error
Error de paridad en RS-232-C.

(14) Format failed
Ha fallado un intento de dar formato, debido posiblemente a un medio defectuoso.

(15) Bad parameter
Hay un error en la lista de paràmetros de un sistema o en una llamada a un procedimiento o funciòn del SuperBASIC.
Se ha hecho un intento de leer datos de un dispositivo de escritura solamente.

(16) Bad medium
El medio està posiblemente defectuoso.

(17) Error in expresiòn
Se ha detectado un error mientras se evaluaba una expresiòn.

(18) Overflow
Revasamiento aritmètico, divisiòn por cero, raìz cuadrada de un nùmero negatico, etc.

(20) Read only
Se ha intentado escribir en un dispositivo compartido.

(21) Bad line
Ha ocurrido un error de sintaxis de SuperBASIC.
Este error puede ocurrir tambièn despuès de un error en una funciòn de una llamada a un procedimiento. Esto previene que se pueda modificar el programa con las lìneas que se estàn introduciendo. Tecleando NEW o CLEAR se modifica esta condiciòn.

Recuperaciòn de Errores

Despuès de un, error se puede rearrancar el programa en la siguiente lìnea mediante el comando:

CONTINUE

Si la condiciòn de error puede ser corregida sin cambiar el programa, se puede rearrancar en la lìnea en la que se ha detectado el error, mediante el comando:

RETRY

Apèndice Seis - Glosario de Tèrminos Informàticos

Accumulator - Parte de la unidad lògica del ordenador que almacena los resultados intermedios de las operaciones.

Address - Un nùmero que direcciona una posiciòn de la memoria del ordenador, o de un dispositivo, donde se almacena informaciòn.

Algorithm - La secuencia de pasos usados para resolver un problema.

Alphanumeric - Generalmente se usa para describir un teclado, y significa que el teclado tiene teclas alfabèticas y numèricas. Un teclado numèrico es el que solamente tiene teclas con los dìgitos del uno al nueve, con algunas teclas adicionales para operaciones aritmèticas, es muy parecido al teclado de una calculadora.

APL - Son las iniciales de 'Automatic Programming/Language' (Lenguaje de Programaciòn Automàtica), un lenguaje desarrollado en los primeros años de la dècada de los sesenta, que soporta una gran cantidad de operadores y estructuras de datos. Usa un juego de caracteres no estàndar.

Application Software - Son grupos de programas preparados para una tarea especìfica, como un proceso de textos o para manejar una lista de correo.

Inteligencia Artificial - La secciòn de la ciencia del ordenador que concentra y desarrolla un comportamiento de la màquina que, desde el punto de vista humano, se podrìa llamar inteligente. Uno de los objetivos de la Inteligencia Artificial es hacer màs ùtiles los ordenadores. La investigaciòn sobre la Inteligencia Artificial nos ayuda a entender mejor nuestros propios procesos del pensamiento.

ASCII - Son las iniciales de American Standard Code for Information Exchange. Es un còdigo casi universal de letras, nùmeros y sìmbolos, que tiene asignado a cada uno de ellos un còdigo del 0 al 255, como el 65 para la letra A.

Assembler - Es un programa que convierte otro programa escrito en lenguaje ensamblador (que es un programa de ordenador en el que una instrucciòn simple, como ADD, se convierte en una instrucciòn simple del ordenador)en un lenguaje que pueda usar directamente el ordenador.

BASIC - Es la abreviatura de 'Beginner's All-purpose Simbolic Instruction Code', el lenguaje màs usado por los microprocesadores. Es muy fàcil de aprender, con muchas de sus sentencias muy parecidas al Inglès. SuperBASIC es una versiòn avanzada del BASIC.

Batch - Un grupo de transacciones que son procesadas por el ordenador en un lote, sin interrupciòn del operador.

Baud - Es la medida de la velocidad de transferencia de datos. Es aproximadamente el nùmero de bits (unidades mìnimas de informaciòn) por segundo.

Benchmark - Es una prueba que se usa para medir algunos aspectos del rendimiento de un ordenador, es el resultado de comparar la ejecuciòn de pruebas similares en diferentes ordenadores.

Binary - Es un sistema numèrico que solamente tiene dos sìmbolos, '0' y '1' (a diferencia del sistema decimal, en el que hay diez sìmbolos, '0', '1', '2', '3', '4', '5', '6', '7', '8' y '9'). Su ordenador 'piensa' en binario.

Boolean Algebra - Es un algebra de toma de decisiones y lògica, desarrollada por el matemàtico inglès George Bool, y la usa el ordenador para tomar sus decisiones. La mayor parte de la potencia de la 'inteligencia' de un ordenador se debe al uso del Algebra de Bool.

Bootstrap - Es un programa que se arranca automaticamente cuando se enciende el ordenador, y que carga a su vez el sistema operativo para que pueda aceptar y entender otros programas.

Buffer - Es un mecanismo de almacenamiento que retiene los datos procedentes de un dispositivo como el teclado, y

que los va liberando a la velocidad que le indica el ordenador.

Bug - Un error en un programa

Bus - Un grupo de conexiones elèctricas usadas para unir un ordenador con un dispositivo auxiliar o con otro ordenador.

Byte - El grupo màs pequeño de bits que compone una palabra del ordenador. Generalmente el ordenador se define como de 'ocho bit' o '16 bit', que significa que la palabra consiste en una combinaciòn de ocho o diez y seis ceros o unos.

Central Processing Unit (CPU) - Es el corazòn del ordenador , donde se efectùan las funciones aritmèticas, lògicas y de control.

Character Code - Es el nùmero que en ASCII (ver ASCII) define un sìmbolo particular, como el 32 para el espacio y el 65 para la letra 'A'.

COBOL - Es la abreviatura de 'Common Business Orientated Language', un lenguaje de programaciòn estàndar, muy parecido al inglès, que se usa principalmente en negocios.

Compiler - Un programa que traduce otro programa escrito en un lenguaje de alto nivel, en lenguaje de màquina que el ordenador es capaz de entender.

Concatenate - Significa añadir (se conoce como concatenaciòn al resultado de añadir dos cadenas).

CP/M - Son las iniciales de 'Control Program/ Microcomputer', un sistema operativo universal de discos desarrollado y comercializado por Digital Research, Pacific Grove, California.

Data - Es el nombre general de la informaciòn que procesa un ordenador.

Database - Es una colecciòn de datos, organizados para permitir un acceso ràpido al ordenador. Una base de datos relaccional es aquella en la que la interconexiòn entre los distintos elementos, dentro de la base de datos, està almacenado explícitamente para ayudar a la manipulaciòn de, y al acceso a, los elementos dentro de dicha base de datos.

Debug - Corregir los errores de un programa.

Disk - Es un medio de almacenamiento magnètico que se usa para almacenar informaciòn y programas del ordenador. El disco se parece a los discos de 45 rpm normales, y los hay de ocho, cinco 1/4 y tres 1/2 pulgadas de diàmetro. Tambièn existen 'microdiscos' para algunos sistemas.

Documentaciòn - Son las instrucciones escritas y explicaciones que suelen acompañar a los programas.

DOS - Son las iniciales de 'Disk Operating System', un programa versàtil que permite al ordenador controlar el sistema de discos.

Dot-matrix printer - Es una impresora que forma las letras y los símbolos por medio de una matriz de puntos, normalmente de ocho por ocho, o siete por cinco.

Double-density - Adjetivo que se usa para describir los discos que usan una tècnica especial para grabar los datos, que , como sugiere su nombre, duplica la capacidad de almacenamiento que tiene un disco.

Dynamic memory - Es un tipo de memoria de ordenador que requiere una recarga constante para conservar su contenido.

EPROM - Son las iniciales de 'Erasable Programmable Read Only Memory' (memoria solo de lectura, programable y borrable), es un dispositivo que contiene informaciòn del ordenador en forma semi-permanente, y que necesita una exposiciòn prolongada a los rayos ultravioletas, para borrar su contenido.

Error messages - Informaciòn que da el ordenador al

usuario, algunas veces consta solamente de nùmeros o unas pocas letras, pero generalmente es una frase (como 'Out of memory') que apunta a un error de programa o de operaciòn que ha forzado a interrumpir la ejecuciòn del programa.

Expert system - Es un programa de ordenador que, trabajando con la experiencia codificada de los expertos humanos, realiza tareas especializadas tan bien como (y, en algunos casos, mejor) lo podrìan hacer los propios expertos humanos. Suelen usar extensas bases de datos de conocimientos, y algunas veces se les llama sistemas basados en el conocimiento.

Field - Es una colecciòn de caracteres que forman un grupo diferenciado, como un còdigo de identificaciòn, un nombre o una fecha; un 'field'(campo), suele formar parte de un registro.

File - Es un grupo de registros relacionados que se procesan unidos, como un fichero de inventario o un fichero de estudiantes.

Firmware - A los componentes sòlidos de un ordenador se les suele llamar 'hardware', a los programas, en forma comprensible para la màquina en un disco o cassette, se les llama 'software', y a los programas que estàn codificados por medio de cables y circuitos, se les llama 'firmware'. En determinadas circunstancias, el 'firmware' puede ser alterado por el 'software', dentro de ciertos lìmites.

Flag - Es un indicador dentro de un programa. El estado del 'flag' nos da informaciòn concerniente a una condiciòn particular.

Floppy disk - Ver 'Disk'

Flowchart - Es una representaciòn de la estructura del programa y su flujo, usando varios tipos de figuras, como un rectàngulo para las acciones del ordenador y un rombo para las decisiones. El 'flowchart' o diagrama de flujo se suele escribir antes de desarrollar el programa.

FORTRAN - Es un lenguaje de alto nivel que se usa generalmente para trabajos científicos (viene de 'FORmula TRANslation).

Gate - Un componente del ordenador que realiza decisiones, permitiendo a un circuito que se dirija en un sentido o en otro, dependiendo de las condiciones satisfechas.

GIGO - Contracción de 'Garbage In Garbage Out', que significa que si entran en el ordenador datos erróneos o morralla, el resultado de procesarlo debe ser también morralla.

Global - A un juego de condiciones que afecta al programa entero, se le llama 'global', lo contrario es 'local'.

Graphics - se emplea este término para todo dato que sale del ordenador que no es alfanumérico o simbólico.

Hardware - Lo componen los componentes físicos del ordenador (ver 'software' y 'firmware').

Heuristic - Método en el que se recurre a tanteos o aproximaciones sucesivas para la resolución de un problema. Se caracteriza por la necesidad de analizar, de forma continuada, cada una de las etapas intermedias sucesivas que señalan la progresión gradual hacia el resultado final, análisis que permite determinar cual ha de ser la etapa o paso siguiente. Esta exigencia de evaluación de las etapas es la que distingue al método heurístico del estocástico.

Hexadecimal - Es un sistema numérico que se suele usar por los programadores de código máquina ya que está íntimamente relacionado al sistema que usan los ordenadores para almacenar datos. Está basado en el número 16 (en contraste con el sistema numérico ordinario que está basado en el 10)

Hex pad - Un teclado, parecido al de una calculadora, que se usa para introducir directamente números hexadecimales.

High-level languages - Lenguajes de programación que son parecidos al inglés. Los lenguajes de bajo nivel son más

parecidos a los que usan los ordenadores. Debido a la necesidad de compilar los lenguajes de alto nivel para que el ordenador los pueda entender, estos lenguajes suelen ser màs lentos que los de bajo nivel.

Human interface - Es la parte externa y visible de un sistema experto, con el que se relacciona el usuario humano; suele funcionar con un lenguaje natural.

Inference system - Es el mecanismo mediante el cual un programa de ordenador llega a conclusiones; algunos ordenadores toman decisiones ràpidas de SI/NO, otros operan en el mundo de la lògica aproximada, donde estàn permitidos ciertos grados de incertidumbre.

Input - Es cualquier informaciòn que sea introducida dentro de un programa durante su ejecuciòn.

I/O - Es la abreviatura de 'Input/Output port' (puerta de entrada/salida), un dispositivo que usa el ordenador para comunicarse con el mundo exterior.

Instruction - Un elemento de còdigo de un programa, que le dice al ordenador que realize una tarea especìfica. Una instrucciòn de lenguaje ensamblador, por ejemplo, puede ser ADD, que le dice al ordenador que realice una suma.

Interpreter - Convierte un programa en lenguaje de alto nivel,en una forma en la que pueda entenderlo el ordenador.

Joystick - Un dispositivo analògico, que introduce señales al ordenador en relacciòn con la posiciòn que ocupa el 'joystick'; se usa generalmente en los programas de juegos.

Kilobyte - Es una unidad de medida de memoria; un kilobyte (generalmente abreviado a K) es igual a 1,024 octetos.

Knowledge base - Es el conocimiento acumulado sobre el que hace sus juicios un sistema experto.

Knowledge engineering - Es el proceso mediante el que se transfiere la experiencia humana a un sistema experto.

Knowledge Information Processing Systema - Esta es la quinta generación de sistemas de ordenadores y se está desarrollando en Japón.

Low-level language - Un lenguaje muy parecido al que usa el ordenador (ver 'High-level language).

Machine language - Es un paso inferior al lenguaje de bajo nivel; es el lenguaje que entiende directamente el ordenador.

Memory - Es el dispositivo o dispositivos que usa el ordenador para guardar la información y los programas que está procesando, y para el juego fijo de instrucciones que le dicen cómo debe ejecutar las demandas del programa. Hay dos tipos básicos de memoria (RAM y ROM).

Microprocessor - Es el corazón del ordenador. Es el 'chip' que 'piensa' y ejecuta las instrucciones del programa.

Modem - Es la abreviatura de MOdulador/DEModulador, un dispositivo que permite a un ordenador comunicarse con otro por una línea telefónica.

Monitor - (a) Una pantalla tipo televisión que usa el ordenador para mostrar resultados; no contiene sintonizador como un televisor; (b) La información, dentro del ordenador que le permite entender y ejecutar las instrucciones de un programa.

Motherboard - Una unidad, generalmente externa, que tiene unas ranuras con contactos que permiten conectar tarjetas con circuitos al ordenador para proporcionar facilidades (como gráficos de alta resolución o control de robots).

Mouse - Una unidad de control, ligeramente más pequeña que una caja de cigarrilos, que rueda sobre la mesa, moviendo en la pantalla un cursor para seleccionar opciones y tomar decisiones dentro de un programa. Los 'ratones' trabajan mediante el sentido de la acción de sus ruedas o leyendo una parrilla modelo sobre la superficie por la que se mueven.

Network - Es un grupo de ordenadores trabajando en paralelo.

Numeric pad - Un dispositivo usado principalmente para introducir informaciòn numèrica dentro del ordenador, es similar al teclado de una calculadora.

Octal - Es un sistema numèrico basado en el ocho (usa los dígitos 0, 1, 2, 3, 4, 5, 6 y 7).

On-line - Estado de un dispositivo que se encuentra bajo control directo del ordenador.

Operating system - Es el programa 'director' o la serie de programas dentro del ordenador que controlan sus operaciones, haciendo cosas como llamar rutinas cuando se necesitan y asignar prioridades.

Output - Cualquier dato producido por el ordenador mientras està en proceso, ya se muestren estos datos en la pantalla o se saquen por la impresora, o sean usados internamente.

Pascal - Un lenguaje de alto nivel, desarrollado a finales de los sesenta por Niklaus Wirth, que fomenta la programaciòn estructurada.

Port - Un 'agujero' de entrada o salida del ordenador, a travès del cual se trasfieren los datos.

Program - La serie de instrucciones que sigue el ordenador para realizar una tarea predeterminada.

PILOT - Un lenguaje de alto nivel, usado generalmente para desarrollar programas educativos.

RAM - Son las iniciales de 'Random Acces Memory'; la memoria principal del ordenador que contiene el programa que se està ejecutando. El contenido de la RAM se puede cambiar, mientras que el contenido de la ROM (Read Only Memory) no se puede cambiar bajo el control de un programa.

Real-time - Cuando està progresando un acontecimiento con

el tiempo en el 'mundo real', se dice que el acontecimiento ocurre en tiempo real. Un ejemplo podría ser un programa que mostrara el desarrollo de una colonia de bacterias a la misma velocidad que lo haría una colonia real. Existen muchos juegos que requieren reacciones en tiempo real. La mayoría de los juegos 'arcade' ocurren en tiempo real.

Refresh - El contenido de las memorias dinàmicas (ver 'memory') necesitan recibir una tensiòn electrica periodica para que puedan mantener su contenido. La señal que 'refresca' la memoria se llama señal de 'refresh'.

Register - Es un lugar de la memoria del ordenador que guarda datos.

Representation - Es la organizaciòn que guarda la información dentro del ordenador de forma que pueda ser manejada bajo el control del sistema base.

Reset - Una señal que devuelve al ordenador al punto en que estaba cuando se encendiò.

ROM - Ver RAM.

RS-232 - Un interface (definido por la Electronic Industries Asociation), que conecta un modem con el equipo y los terminales asociados al ordenador; el QL tiene dos puertas de salida RS-232-C.

S-100 bus - Es tambièn un interface estàndar (ver RS-232) hecho de 100 lîneas comunes de comunicaciòn, que se usan para conectar tarjetas de circuitos dentro de los micro-ordenadores.

SNOBOL - Un lenguaje de alto nivel, desarrollado por Bell Laboratories, que usa patròn de reconocimiento y manipulaciòn de cadenas.

Software - Es el programa que sigue el ordenador (ver mfirmware').

Stack - El punto final de una serie de eventos que son

accedidos en forma de 'ùltimo en entrar primero en salir'.

Subroutine - Un bloque de còdigo o programa, que es llamado varias veces dentro de otro programa.

Symbolic inference - Ver 'inference system'.

Sintax - Es el conjunto de las reglas de estructuras que gobierna el lenguaje del ordenador.

Systems software - Secciones de còdigo que realizan tareas administrativas, o asisten a la escritura de otros programas, pero que no se usan para realizar la tarea final del ordenador.

Thermal printer - Un dispositivo que imprime los datos de salida del ordenador sobre papel con sensibilizaciòn tèrmica.

Time-sharing - Este tèrmino se usa para referirse a un gran nùmero de usuarios, en terminales independientes, que hacen uso de un solo ordenador, que divide su tiempo entre los usuarios, de forma que cada uno de ellos parece que tiene la total atenciòn del ordenador.

Turnkey system - Es un sistema de ordenador (generalmente para uso de negocios) que se vende preparado para trabajar, mediante la sola acciòn de un giro de llave.

VLSI - Very Large Scale Integration. Es la integraciòn, a gran escala de los componentes de un 'chip', que se està desarrollando en Japòn y en Estados Unidos y que puede contener hasta 10 millones de transistores por 'chip' (los actuales contienen el equivalente a medio millòn de transistores como mucho).

Volatile Memory - Un dispositivo de memoria que pierde su contenido cuando se le corta la corriente.

Word processor - Es un ordenador dedicado (o un ordenador ejecutando un programa de 'word processor'), que da acceso a una 'màquina de escribir inteligente'.

Apêndice Siete -
El Grupo Independiente de Usuarios de QL (IQLUG)

Este es un grupo muy activo, que publica la revista mensual *Quanta*. El grupo me permitiò amablemente incluir en este libro algunos artículos de su revista. La procedencia se explica en cada uno de ellos.

El director y editor de la revista es Leon Heller, que tiene una gran experiencia con el 68000 y el 68006, con una gran cantidad de dispositivos que usan este procesador. Tiene funcionando un ensamblador de 68000 en uno de sus sistemas, que le permite escribir còdigo màquina del 68000 y ejecutarlo en el 68000 y el 68008.

El co-fundador del IQLUG es Brian Pain, el secretario, y se le puede encontrar en 24 Oxford Street, Stony Stratford, Milton Keynes. Puede proporcionar informaciòn del club y de *Quanta*. Brian da conferencias en un colegio de educaciòn avanzada, y formò, a finales de los setenta, lo que es ahora el Grupo Nacional de Usuarios del TRS-80 y del Genie.

La incorporaciòn como miembro al IQLUG se hace mediante la subscripciòn a la revista y està abierta a todo aquel que tenga interès por el QL. El club asiste a los miembros en los problemas relacionados con el QL. Se hacen seminarios periòdicos alrededor del Reino Unido. El club mantiene una librería de programas disponible para los miembros.

Apêndice Ocho - Conexionado

Cuando desempaquete su QL, podrà ver el ordenador con su guìa del usuario, una fuente de alimentaciòn, varios cables, el cable de la red de datos, dos estuches de Microdrives (uno con cuatro Microdrives sin grabar y el otro con los Microdrives de los programas que se suministran con el QL) y unas pequeñas patas de plàstico para poner bajo el QL por si se quiere variar el àngulo del teclado. Probablemente encontrarà - como yo lo hice - que es màs fàcil de manejar el QL si le coloca estas patas.

Ocultos en la parte de atràs de su QL se encuentran unas ranuras y zòcalos , donde se conectan los cartuchos ROM, impresoras, 'joysticks' y otros dispositivos.

Si gira el QL para ver èstos conectores, verà que son para los siguientes usos (empezando por el extremo izquierdo segùn se ve desde la parte de atràs de la màquina):

- **red de datos**
- **red de datos**
- **alimentaciòn**
- **salida para monitor RGB**
- **salida para televisor UHF**
- **puerta serie uno**
- **puerta serie dos**
- **control uno (joystick)**
- **control dos**
- **cartuchos ROM**

Hay una ranura de expansiòn en el lado derecho (mirando desde atràs) para conectar una expansiòn de memoria de medio megabyte u otros perifèricos.

Las ranuras de la red de datos le permiten conectar entre sì hasta 64 QL's y Spectrums en lo que se llama una 'local area network' (red de datos de àrea local, QLAN en el lenguaje Sinclair). Los diferentes ordenadores en la red de datos pueden 'hablar' a cada uno de los otros a 100K baudios (baudio es la unidad de velocidad de trasmisiòn de

información y para que se haga una idea de su rapidez, considere que el Spectrum lee los programas del cassette aproximadamente 1.5K). Los datos pueden mandarse por la red de datos de àrea local a cualquier ordenador que estè preparado para recibirlos.

En el frente, a mano derecha, estàn las ranuras para los Microdrives. No introduzca un Microdrive en su ranura antes de que el QL estè conectado. Si lo hace, corre el riesgo de destruir los datos almacenados en ese Microdrive. Cuando un Microdrive està girando y el QL lo està leyendo, la luz que està al lado de su ranura està encendida. No intente sacar el Microdrive mientras la luz estê encendida porque puede dañar el ordenador y el Microdrive.

La puerta RGB es para conectar un monitor RGB. Si solo ha visto funcionar el QL en una televisiòn domèstica, recibirà una agradable sorpresa cuando conecte un monitor RGB. La imàgen es mucho màs clara.

Apêndice Nueve - Lecturas Complementarias

Hay varios libros sobre el QL que le pueden resultar interesantes. Los que he encontrado màs ùtiles son:

An Introduction to Programming the Sinclair QL - R A & J W Penfold (Babani, 0 85934 125 9, £1.95).

Good Programming with QL Superbasic - Roy Atherton (Longman 0 582 29662 5, £5.95).

También pueden ser de interès:

Quill, Easel, Archive and Abacus on the QL - A McCallum-Varey (Sunshine, 0 946408 55 6, £6.95).

Advanced Programming with Sinclair QL - Martin Gandoff (Hutchinson,0 09 158961 4, £6.95).

Entre los libros de interès general estàn:

Tim Hartnell's QL Games Compendium - Tim Hartnell (Interface, 0 947695 04 4, £5.95).

The Art od Structured Programming Peter Juliff (Interface, 0 907563 79 1, £5.95).

NOTAS

NOTAS

De Tim Hartnell, el autor al que la revista Personal Computer World llamó "Mr Sinclair", nos llega este trabajo esencial para los programadores del Sinclair QL.

Paso a paso, elemento por elemento, Tim Hartnell le guía por el SuperBASIC, mostrándole lo excitante e importante que es éste lenguaje. Verá como adaptar sus conocimientos y estilo de programación para usar con ventaja las posibilidades del SuperBASIC. Como cada comando está demostrado con programas, se irá construyendo una librería de inapreciable valor, con juegos y utilidades.

Entre los tesoros de este libro está un reloj que trabaja en multi-tarea escrito en código de máquina, que aparece en la parte inferior derecha de la pantalla, sin interferir en lo que está haciendo. Con el QLogo le dará a su QL una versión del LOGO que proporciona una gran cantidad de comandos LOGO estándar y que se puede hacer funcionar en modo directo. La Máquina FORTH es un mini-FORTH escrito en SuperBASIC que le permitirá aprender este lenguaje por si mismo en el QL, sin tener que comprar un programa aparte.

Hay muchas más cosas en este libro, incluidos varios juegos, una sección dedicada a la creación de aventuras (completada con una aventura de 20K para resolver), un programa para crear modelos de proyecciones financieras, manejo de ficheros y una rutina para definirse sus propios caracteres, así como una introducción al procesador 68008. Todo esto está aquí, en este trabajo esencial.

